CHAPTER 18 INSTRUCTION SET

18.1 Operation List

18.1.1 Operand identifiers and description

In the operand field of each instruction, describe the operands according to the description for the operand identifiers of the instruction. (For details, see the assembler specifications.) Select one of the entries under the description, if present. The uppercase alphabetic characters and the +, -, #, $, !, and [ ] symbols are keywords which should be described exactly as shown.

For immediate data, describe a proper numeric value or label. To describe the immediate data with a label, be sure to describe the symbols #, $, !, [ ] as well.

Table 18-1 Operand Identifiers and Description

Identifier	Description
r	R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15
r1	R0, R1, R2, R3, R4, R5, R6, R7
r2	C, B
rp	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp1	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp2	DE, HL, VP, UP
sfr	Special function register name (See Table 3-4)
sfrp	Special function register name of special function register that can be handled in 16-bit units (See Table 3-4)
post	RP0, RP1, RP2, RP3, RP4, RP5/PSW, RP6, RP7 (more than one can be described, but RP5 can be described only with PUSH and POP instructions and PSW can be described only with PUSHU and POPU instructions)
mem	[DE], [HL], [DE+], [HL+], [DE-], [HL-], [VP], [UP]: Register indirect mode [DE+A], [HL+A], [DE+B], [HL+B], [VP+DE], [VP+HL]: Based index mode [DE+byte], [HL+byte], [VP+byte], [UP+byte], [SP+byte]: Based mode word[A], word[B], word[DE], word[HL]: Indexed mode
saddr	FE20H-FF1FH immediate data or label
saddrp	FE20H-FF1EH immediate data (where bit0=0) or label (16-bit operation)
$addr16	0000H-FDFFH immediate data or label: Relative addressing
!addr16	0000H-FDFFH immediate data or label: Immediate addressing (however, up to FFFFH can be described with MOV instruction)
addr11	800H-FFFH immediate data or label
addr5	40H-7EH immediate data or label (where bit 0=0)(Note) or label
word	16-bit immediate data or label
byte	8-bit immediate data or label
bit	3-bit immediate data or label
n	3-bit immediate data or label (0-7)

Note: Do not make a word access to any odd address (bit 0=1).

Remarks 1: rp and rpl are the same in register names that can be described, but differ in generated codes. (See 18.2.)

2: The addresses of all space can be addressed by using immediate addressing. Relative addressing can be used only to address the range of "top address of the following instruction -128" to "top address +127".

The function names as well as the absolute names (R0-15 and RP0-RP7) can be described in 8-bit register identifiers r and rl and 16-bit register pair identifiers rp, rpl, and post. Tables 9-2 and 9-3 list the correspondence between the absolute and function names of the registers.

Table 18-2 Correspondence between Absolute and Function Names of 8-bit Registers

Absolute name	Function name		Absolute name	Function name	
	RSS=0	RSS=1		RSS=0	RSS=1
R0	X		R8	VP_L	VP_L
R1	A		R9	VP_H	VP_H
R2	C		R10	UP_L	UP_L
R3	B		R11	UP_H	UP_H
R4		X	R12	E	E
R5		A	R13	D	D
R6		C	R14	L	L
R7		B	R15	H	H

Table 18-3 Correspondence between Absolute and Function Names of 16-bit Register Pairs

Absolute name	Function name	
	RSS=0	RSS=1
RP0	AX	
RP1	BC	
RP2		AX
RP3		BC
RP4	VP	VP
RP5	UP	UP
RP6	DE	DE
RP7	HL	HL

The RSS is a register set selection flag (PSW bit 5). The correspondence between the absolute and function names is changed by setting or resettting the flag.

18.1.2 Legend on operation explanation

A	: A register or an 8-bit accumulator
X	: X register
B	: B register
C	: C register
D	: D register
E	: E register
H	: H register
L	: L register
R0-R15	: Registers 0-15 (absolute names)
AX	: Register pair (AX) or a 16-bit accumulator
BC	: Register pair (BC)
DE	: Register pair (DE)
HL	: Register pair (HL)
RP0-RP7	: Register pairs 0-7 (absolute names)
PC	: Program counter
SP	: Stack pointer
UP	: User stack pointer
PSW	: Program status word
CY	: Carry flag
AC	: Auxiliary carry flag
Z	: Zero flag
P/V	: Parity/overflow flag
S	: Sign flag
TPF	: Table position flag
RBS	: Register bank select register
RSS	: Register set select register
IE	: Interrupt enable flag
STBC	: Standby control register
WDM	: Watchdog timer mode register
jdisp8	: Signed 8-bit data (displacement: -128 to +127)
( )	: Contents of memory addressed by the contents of register or address enclosed in parentheses. When (+) or (-) is given, the contents in ( ) are incremented or decremented by one after the instruction is executed.
(( ))	: Contents of memory addressed by the contents of

memory addressed by address enclosed in double parentheses.

xxH : Hexadecimal number

xH, xL : High-order eight bits and low-order eight bits of 16-bit register

18.1.3 Explanation of symbols under column of flags

Table 18-4 Symbols and Explanation under Column of Flags

Symbol	Explanation
(Blank)	No change
0	Reset to 0
1	Set to 1
x	Set or reset according to result
P	P/V flag operates as parity flag
V	P/V flag operates as overflow flag
R	Previously saved value is restored

18.1.4 Operation list of basic instruction

(1) 8-bit data transfer instruction: MOV, XCH

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	rl, #byte	2	rl ← byte					
	saddr, #byte	3	(saddr) ← byte					
	sfr(Note), #byte	3	sfr ← byte					
	r, rl	2	r ← rl					
	A, rl	1	A ← rl					
	A, saddr	2	A ← (saddr)					
	saddr, A	2	(saddr) ← A					
	saddr, saddr	3	(saddr) ← (saddr)					
	A, sfr	2	A ← sfr					
	sfr, A	2	sfr ← A					
	A, mem	1-4	A ← (mem)					
	mem, A	1-4	(mem) ← A					
	A, [saddrp]	2	A ← ((saddrp))					
	[saddrp], A	2	((saddrp)) ← A					
	A, !addr16	4	A ← (addr16)					
	!addr16, A	4	(addr16) ← A					
	PSWL, #byte	3	PSW_L ← byte	x	x	x	x	x
	PSWH, #byte	3	PSW_H ← byte					
	PSWL, A	2	PSW_L ← A	x	x	x	x	x
	PSWH, A	2	PSW_H ← A					
	A, PSWL	2	A ← PSW_L					
	A, PSWH	2	A ← PSW_H					
XCH	A, rl	1	A ↔ rl					
	r, rl	2	r ↔ rl					
	A, mem	2-4	A ↔ (mem)					
	A, saddr	2	A ↔ (saddr)					
	A, sfr	3	A ↔ sfr					
	A, [saddrp]	2	A ↔ ((saddrp))					
	saddr, saddr	3	(saddr) ↔ (saddr)					

Note: If STBC or WDM is described in sfr, the instruction becomes another dedicated instruction and the number of bytes differ from those listed here.

(2) 16-bit data transfer instruction: MOVW, XCHW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVW	rpl, #word	3	rpl ← word					
	saddrp, #word	4	(saddrp) ← word					
	sfrp, #word	4	sfrp ← word					
	rp, rpl	2	rp ← rpl					
	AX, saddrp	2	AX ← (saddrp)					
	saddrp, AX	2	(saddrp) ← AX					
	saddrp, saddrp	3	(saddrp) ← (saddrp)					
	AX, sfrp	2	AX ← sfrp					
	sfrp, AX	2	sfrp ← AX					
	rpl, !addr16	4	rpl ← (addr16)					
	!addr16, rpl	4	(addr16) ← rpl					
	AX, mem	2-4	AX ← mem					
	mem, AX	2-4	mem ← AX					
XCHW	AX, saddrp	2	AX ↔ (saddrp)					
	AX, sfrp	3	AX ↔ sfrp					
	saddrp, saddrp	3	(saddrp) ↔ (saddrp)					
	rp, rpl	2	rp ↔ rpl					
	AX, mem	2-4	AX ↔ mem					

(3) 8-bit arithmetic and logical instruction: ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADD	A, #byte	2	A, CY ← A+byte	x	x	x	V	x
	saddr, #byte	3	(saddr), CY ← (saddr)+byte	x	x	x	V	x
	sfr, #byte	4	sfr, CY ← sfr +byte	x	x	x	V	x
	r, rl	2	r, CY ← r+rl	x	x	x	V	x
	A, saddr	2	A, CY ← A+(saddr)	x	x	x	V	x
	A, sfr	3	A, CY ← A+sfr	x	x	x	V	x
	saddr, saddr	3	(saddr), CY ← (saddr)+(saddr)	x	x	x	V	x
	A, mem	2-4	A, CY ← A+(mem)	x	x	x	V	x
	mem, A	2-4	(mem), CY ← (mem)+A	x	x	x	V	x
ADDC	A, #byte	2	A, CY ← A+byte+CY	x	x	x	V	x
	saddr, #byte	3	(saddr), CY ← (saddr)+byte+CY	x	x	x	V	x
	sfr, #byte	4	sfr, CY ← sfr+byte+CY	x	x	x	V	x
	r, rl	2	r, CY ← r+rl+CY	x	x	x	V	x
	A, saddr	2	A, CY ← A+(saddr)+CY	x	x	x	V	x
	A, sfr	3	A, CY ← A+sfr+CY	x	x	x	V	x
	saddr, saddr	3	(saddr), CY← (saddr)+(saddr)+CY	x	x	x	V	x
	A, mem	2-4	A, CY ← A+ (mem)+CY	x	x	x	V	x
	mem, A	2-4	(mem), CY ← (mem)+A+CY	x	x	x	V	x

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
SUB	A, #byte	2	A, CY ← A-byte	x	x	x	V	x
	saddr, #byte	3	(saddr), CY ← (saddr)-byte	x	x	x	V	x
	sfr, #byte	4	sfr, CY ← sfr-byte	x	x	x	V	x
	r, rl	2	r, CY ← r-rl	x	x	x	V	x
	A, saddr	2	A, CY ← A- (saddr)	x	x	x	V	x
	A, sfr	3	A, CY ← A- sfr	x	x	x	V	x
	saddr, saddr	3	(saddr), CY ← (saddr)-(saddr)	x	x	x	V	x
	A, mem	2-4	A, CY ← A- (mem)	x	x	x	V	x
	mem, A	2-4	(mem), CY ← (mem)-A	x	x	x	V	x
SUBC	A, #byte	2	A, CY ← A-byte-CY	x	x	x	V	x
	saddr, #byte	3	(saddr), CY ← (saddr)-byte-CY	x	x	x	V	x
	sfr, #byte	4	sfr, CY ← sfr-byte-CY	x	x	x	V	x
	r, rl	2	r, CY ← r-rl-CY	x	x	x	V	x
	A, saddr	2	A, CY ← A-(saddr)-CY	x	x	x	V	x
	A, sfr	3	A, CY ← A-sfr-CY	x	x	x	V	x
	saddr, saddr	3	(saddr), CY ← (saddr)-(saddr)-CY	x	x	x	V	x
	A, mem	2-4	A, CY ← A- (mem)-CY	x	x	x	V	x
	mem, A	2-4	(mem), CY ← (mem)-A-CY	x	x	x	V	x
AND	A, #byte	2	A ← A ∧ byte		x	x	P	P
	saddr, #byte	3	(saddr) ← (saddr)∧ byte		x	x	P	P
	sfr, #byte	4	sfr ← sfr∧ byte		x	x	P	P
	r, rl	2	r← r ∧ rl		x	x	P	P
	A, saddr	2	A ← A ∧ (saddr)		x	x	P	P
	A, sfr	3	A ← A∧sfr		x	x	P	P
	saddr, saddr	3	(saddr) ← (saddr)∧ (saddr)		x	x	P	P
	A, mem	2-4	A← A∧(mem)		x	x	P	P
	mem, A	2-4	(mem) ← (mem) A		x	x	P	P

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
OR	A, #byte	2	A ← A V byte	x	x		P	
	saddr, #byte	3	(saddr) ← (saddr) V byte		x	x	P	
	sfr, #byte	4	sfr ← sfr V byte		x	x	P	
	r, rl	2	r ← r V rl		x	x	P	
	A, saddr	2	A ← V (saddr)		x	x	P	
	A, sfr	3	A ← A V sfr		x	x	P	
	saddr, saddr	3	(saddr) ← (saddr) V (saddr)		x	x	P	
	A, mem	2-4	A ← A V (mem)		x	x	P	
	mem, A	2-4	(mem) ← (mem) V A		x	x	P	
XOR	A, #byte	2	A← A V byte	x	x		P	
	saddr, #byte	3	(saddr) ← (saddr) V byte	x	x		P	
	sfr, #byte	4	sfr ← sfr V byte	x	x		P	
	r, rl	2	r ← r V rl	x	x		P	
	A, saddr	2	A ← A V (saddr)	x	x		P	
	A, sfr	3	A ← A V sfr	x	x		P	
	saddr, saddr	3	(saddr) ← (saddr) V (saddr)	x	x		P	
	A, mem	2-4	A ← A V (mem)	x	x		P	
	mem, A	2-4	(mem) ← (mem) V A	x	x		P	
CMP	A, #byte	2	A-byte	x	x	x	V	x
	saddr, #byte	3	(saddr)-byte	x	x	x	V	x
	sfr, #byte	4	sfr-byte	x	x	x	V	x
	r, rl	2	r-rl	x	x	x	V	x
	A, saddr	2	A-(saddr)	x	x	x	V	x
	A, sfr	3	A-sfr	x	x	x	V	x
	saddr, saddr	3	(saddr)-(saddr)	x	x	x	V	x
	A, mem	2-4	A-(mem)	x	x	x	V	x
	mem, A	2-4	(mem)-A	x	x	x	V	x

(4) 16-bit arithmetic and logical instruction: ADDW, SUBW, CMPW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDW	AX, #word	3	AX, CY ← AX+word	x	x	x	V	x
	saddrp, #word	4	(saddrp), CY ← (saddrp)+word	x	x	x	V	x
	sfrp, #word	5	sfrp, CY ← sfrp+word	x	x	x	V	x
	rp, rp1	2	rp, CY ← rp+rp1	x	x	x	V	x
	AX, saddrp	2	AX, CY ← AX+(saddrp)	x	x	x	V	x
	AX, sfrp	3	AX, CY ← AX+sfrp	x	x	x	V	x
	saddrp, saddrp	3	(saddrp),CY ← (saddrp)+(saddrp)	x	x	x	V	x
SUBW	AX, #word	3	AX, CY ← AX-word	x	x	x	V	x
	saddrp, #word	4	(saddrp), CY ← (saddrp)-word	x	x	x	V	x
	sfrp, #word	5	sfrp, CY ← sfrp-word	x	x	x	V	x
	rp, rp1	2	rp, CY ← rp-rp1	x	x	x	V	x
	AX, saddrp	2	AX, CY ← AX-(saddrp)	x	x	x	V	x
	AX, sfrp	3	AX, CY ← AX-sfrp	x	x	x	V	x
	saddrp, saddrp	3	(saddrp), CY ← (saddrp)-(saddrp)	x	x	x	V	x
CMPW	AX, #word	3	AX-word	x	x	x	V	x
	saddrp, #word	4	(saddrp)-word	x	x	x	V	x
	sfrp, #word	5	sfrp-word	x	x	x	V	x
	rp, rp1	2	rp-rp1	x	x	x	V	x
	AX, saddrp	2	AX-(saddrp)	x	x	x	V	x
	AX, sfrp	3	AX-sfrp	x	x	x	V	x
	saddrp, saddrp	3	(saddrp)-(saddrp)	x	x	x	V	x

(5) Multiplication and division instruction: MULU, DIVUW, MULUW, DIVUX

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MULU	rl	2	AX ← Axrl					
DIVUW	rl	2	AX (quotient), rl(remainder) ← AX÷rl					
MULUW	rpl	2	AX (high-order 16 bits), rpl (low-order 16 bits) ← AXxrpl					
DIVUX	rpl	2	AXDE (quotient), rpl (remainder) ← AXDE÷rpl					

(6) Signed multiplication: MULW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MULW	rp1	2	AX (high-order 16 bits), rp1 (low-order 16 bits) ← AXxrp1					

(7) Increment and decrement instruction: INC, DEC, INCW, DECW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
INC	r1	1	r1 ← r1+1	x	x	x	V	
	saddr	2	(saddr) ← (saddr)+1	x	x	x	V	
DEC	r1	1	r1 ← r1-1	x	x	x	V	
	saddr	2	(saddr) ← (saddr) -1	x	x	x	V	
INCW	rp2	1	rp2 ← rp2+1					
	saddrp	3	(saddrp) ← (saddrp) +1					
DECW	rp2	1	rp2 ← rp2-1					
	saddrp	3	(saddrp) ← (saddrp) -1					

(8) Shift and rotata instruction: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ROR	rl, n	2	$(CY, rl_7 \leftarrow rl_0, rl_{m-1} \leftarrow rl_m) \times n$ n=0-7				P	x
ROL	rl, n	2	$(CY, rl_0 \leftarrow rl_7, rl_{m+1} \leftarrow rl_m) \times n$ n=0-7				P	x
RORC	rl, n	2	$(CY \leftarrow rl_0, rl_7 \leftarrow CY, rl_{m-1} \leftarrow rl_m) \times n$ n=0-7				P	x
ROLC	rl, n	2	$(CY \leftarrow rl_7, rl_0 \leftarrow CY, rl_{m+1} \leftarrow rl_m) \times n$ n=0-7				P	x
SHR	rl, n	2	$(CY \leftarrow rl_0, rl_7 \leftarrow 0, rl_{m-1} \leftarrow rl_m) \times n$ n=0-7	x	x	0	P	x
SHL	rl, n	2	$(CY \leftarrow rl_7, rl_0 \leftarrow 0, rl_{m+1} \leftarrow rl_m) \times n$ n=0-7	x	x	0	P	x
SHRW	rl, n	2	$(CY \leftarrow rpl_0, rpl_{15} \leftarrow 0, rpl_{m-1} \leftarrow rpl_m) \times n$ n=0-7	x	x	0	P	x
SHLW	rpl, n	2	$(CY \leftarrow rpl_{15}, rpl_0 \leftarrow 0, rpl_{m+1} \leftarrow rpl_m) \times n$ n=0-7	x	x	0	P	x
ROR4	[rpl]	2	$A_{3-0} \leftarrow (rpl)_{3-0}, (rpl)_{7-4} \leftarrow A_{3-0}, (rpl)_{3-0} \leftarrow (rpl)_{7-4}$					
ROL4	[rpl]	2	$A_{3-0} \leftarrow (rpl)_{7-4}, (rpl)_{3-0} \leftarrow A_{3-0}, (rpl)_{7-4} \leftarrow (rpl)_{3-0}$					

Remarks: n under the shift or rotate instruction indicates the shift or rotate count.

(9) BCD adjustment instruction: ADJBA, ADJBS

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADJBA		2	Decimal Adjust Accumulator	x	x	x	P	x
ADJBS								

(10) Data conversion instruction: CVTBW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CVTBW		1	When A_7=0 X ← A, A← 00H When A_7=1 X ← A, A← FFH					

(11) Bit manipulation instruction: MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1 (1/2)

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV1	CY, saddr. bit	3	CY ← (saddr.bit)					x
	CY, sfr.bit	3	CY ← sfr.bit					x
	CY, A. bit	2	CY ← A.bit					x
	CY, X. bit	2	CY ← X.bit					x
	CY, PSWH.bit	2	CY ← PSW_H.bit					x
	CY, PSWL.bit	2	CY ← PSW_L.bit					x
	saddr.bit, CY	3	(saddr.bit) ← CY					
	sfr.bit, CY	3	sfr.bit ← CY					
	A.bit, CY	2	A.bit ← CY					
	X.bit, CY	2	X.bit ← CY					
	PSWH.bit, CY	2	PSW_H.bit ← CY					
	PSWL.bit, CY	2	PSW_L.bit ← CY					
AND1	CY, saddr.bit	3	CY ← CY ∧ (saddr.bit)					x
	CY,/saddr.bit	3	CY ← CY ∧ $\overline{(saddr.bit)}$					x
	CY, sfr.bit	3	CY ← CY ∧ sfr.bit					x
	CY,/sfr.bit	3	CY ← CY ∧ $\overline{sfr.bit}$					x
	CY, A.bit	2	CY ← CY ∧ A.bit					x
	CY,/A.bit	2	CY ← CY ∧ $\overline{A.bit}$					x
	CY, X.bit	2	CY ← CY ∧ X.bit					x
	CY,/X.bit	2	CY ← CY ∧ $\overline{X.bit}$					x
	CY,PSWH.bit	2	CY ← CY ∧ PSW_H.bit					x
	CY,/PSWH.bit	2	CY ← CY ∧ $\overline{PSW_H.bit}$					x
	CY, PSWL.bit	2	CY ← CY ∧ PSW_L.bit					x
	CY,/ PSWL.bit	2	CY ← CY ∧ $\overline{PSW_L.bit}$					x
OR1	CY, saddr.bit	3	CY ← CY ∨ (saddr.bit)					x
	CY,/saddr.bit	3	CY ← CY ∨ $\overline{(saddr.bit)}$					x
	CY, sfr.bit	3	CY ← CY ∨ sfr.bit					x
	CY,/sfr.bit	3	CY ← CY ∨ $\overline{sfr.bit}$					x
	CY, A.bit	2	CY ← CY ∨ A.bit					x
	CY,/A.bit	2	CY ← CY ∨ $\overline{A.bit}$					x
	CY, X.bit	2	CY ← CY ∨ X.bit					x
	CY,/X.bit	2	CY ← CY ∨ $\overline{X.bit}$					x
	CY,PSWH.bit	2	CY ← CY ∨ PSW_H.bit					x
	CY,/PSWH.bit	2	CY ← CY ∨ $\overline{PSW_H.bit}$					x
	CY, PSWL.bit	2	CY ← CY ∨ PSW_L.bit					x
	CY, PSWL.bit	2	CY ← CY ∨ $\overline{PSW_L.bit}$					x

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR1	CY, sddr. bit	3	CY ← CY ∨ (saddr. bit)					x
	CY, sfr.bit	3	CY ← CY ∨ sfr.bit					x
	CY, A. bit	2	CY ← CY ∨ A.bit					x
	CY, X. bit	2	CY ← CY ∨ X.bit					x
	CY, PSWH.bit	2	CY ← CY ∨ PSW_H.bit					x
	CY, PSWL.bit	2	CY ← CY ∨ PSW_L.bit					x
SET1	saddr.bit	2	(saddr.bit) ← 1					
	sfr.bit	3	sfr.bit ← 1					
	A.bit	2	A.bit ← 1					
	X.bit	2	X.bit ← 1					
	PSWH.bit	2	PSW_H.bit ← 1					
	PSWL.bit	2	PSW_L.bit ← 1	x	x	x	x	x
	CY	1	CY ← 1					1
CLR1	saddr.bit	2	(saddr.bit) ← 0					
	sfr.bit	3	sfr.bit← 0					
	A.bit	2	A.bit ← 0					
	X.bit	2	X.bit ← 0					
	PSWH.bit	2	PSW_H.bit ← 0					
	PSWL.bit	2	PSW_L.bit ← 0	x	x	x	x	x
	CY	1	CY ← 0					0
NOT1	saddr.bit	3	(saddr.bit) ← $\overline{(saddr.bit)}$					
	sfr.bit	3	sfr.bit ← $\overline{sfr.bit}$					
	A.bit	2	A.bit ← $\overline{A.bit}$					
	X.bit	2	X.bit ← $\overline{X.bit}$					
	PSWH.bit	2	PSW_H.bit ← $\overline{PSW_H.bit}$					
	PSWL.bit	2	PSW_L.bit ← $\overline{PSW_L.bit}$	x	x	x	x	x
	CY	1	CY ← $\overline{CY}$					x

(12) Call and return instruction: CALL, CALLF, CALLT, BRK, RET, RETB, RETI

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CALL	!addr16	3	$(SP-1) \leftarrow (PC+3)_H$, $(SP-2) \leftarrow (PC+3)_L$, PC ← addr16, SP ← SP-2					
	rp1	2	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$, $PC_H \leftarrow rp1_H$, $PC_L \leftarrow rp1_L$, SP ← SP-2					
	[rp1]	2	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$, $PC_H \leftarrow (rp1+1)$, $PC_L \leftarrow (rp1)$, SP ← SP-2					
CALLF	!addr11	2	$(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$, $PC_{15-11} \leftarrow 00001$, $PC_{10-0} \leftarrow$ addr11, SP ← SP-2					
CALLT	[addr5]	1	$(SP-1) \leftarrow (PC+1)_H$, $(SP-2) \leftarrow (PC+1)_L$, $PC_H \leftarrow$ (TPF, 00000000, addr5+1), $PC_L \leftarrow$ (TPF, 00000000, addr5), SP ← SP-2					
BRK		1	$(SP-1) \leftarrow PSW_H$, $(SP-2) \leftarrow PSW_L$, $(SP-3) \leftarrow (PC+1)_H$, $(SP-4) \leftarrow (PC+1)_L$, $PC_L \leftarrow$ (003EH), $PC_H \leftarrow$ (003FH), SP ← SP-4, IE ← 0					
RET		1	$PC_L \leftarrow$ (SP), $PC_H \leftarrow$ (SP+1), SP ← SP+2					
RETB		1	$PC_L \leftarrow$ (SP), $PC_H \leftarrow$ (SP+1), $PSW_L \leftarrow$ (SP+2), $PSW_H \leftarrow$ (SP+3) SP ← SP+4	R	R	R	R	R
RETI		1	$PC_L \leftarrow$ (SP), $PC_H \leftarrow$ (SP+1), $PSW_L \leftarrow$ (SP+2), $PSW_H \leftarrow$ (SP+3) SP ← SP+4	R	R	R	R	R

(13) Stack handling instruction: PUSH, PUSHU, POP, POPU, MOVW, INCW, DECW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
PUSH	sfrp	3	(SP-1) ← sfr_H, (SP-2) ← sfr_L, SP ← SP-2					
	post	2	{(SP-1) ← $post_H$, (SP-2) ← $post_L$, SP ← SP-2}xn					
	PSW	1	(SP-1) ← PSW_H, (SP-2) ← PSW_L, SP ← SP-2					
PUSHU	post	2	{(UP-1) ← $post_H$, (UP-2) ← $post_L$, UP ← UP-2}xn					
POP	sfrp	3	sfr_L ← (SP), sfr_H ← (SP+1), SP ← SP+2					
	post	2	{$post_L$ ← (SP), $post_H$ ← (SP+1), SP ← SP+2}xn					
	PSW	1	PSW_L ← (SP), PSW_H ← (SP+1), SP ← SP+2	R	R	R	R	R
POPU	post	2	{$post_L$ ← (UP), $post_H$ ← (UP+1), UP ← UP+2}xn					
MOVW	SP, #word	4	SP ← word					
	SP, AX	2	SP ← AX					
	AX, SP	2	AX ← SP					
INCW	SP	2	SP ← SP+1					
DWCW	SP	2	SP ← SP-1					

Remarks: n under the stack handling instruction indicates the number of registers described as post.

(14) Special instruction: CHKL, CHKLA

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CHKL	sfr	3	(pin level) V (signal level at prestage of output buffer)	x	x		P	
CHKLA	sfr	3	A ← (pin level) V (signal level at prestage of output buffer)	x	x		P	

(15) Unconditional branch instruction: BR

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BR	!addr16	3	PC ← addr16					
	rpl	2	$PC_H \leftarrow rpl_H$, $PC_L \leftarrow rpl_L$					
	[rpl]	2	$PC_H \leftarrow (rpl+1)$, $PC_L \leftarrow (rpl)$					
	$ addr16	2	PC ← PC+2+jdisp8					

(16) Conditional branch instruction: BC, BL, BNC, BNL, BZ, BE, BNZ, BNE, BV, BPE, BNV, BPO, BN, BP, BGT, BGE, BLT, BLE, BH, BNH, BT, BF, BTCLR, BFSET, DBNZ

(1/2)

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BC BL	$ addr16	2	PC ← PC+2+jdisp8 if CY=1					
BNC BNL	$ addr16	2	PC ← PC+2+jdisp8 if CY=0					
BZ BE	$ addr16	2	PC ← PC+2+jdisp8 if Z=1					
BNZ BNE	$ addr16	2	PC ← PC+2+jdisp8 if Z=0					
BV BPE	$ addr16	2	PC ← PC+2+jdisp8 if P/V=1					
BNV BPO	$ addr16	2	PC ← PC+2+jdisp8 if P/V=0					
BN	$addr16	2	PC ← PC+2+jdisp8 if S=1					
BP	$addr16	2	PC ← PC+2+jdisp8 if S=0					
BGT	$addr16	3	PC ← PC+3+jdisp8 if (P/V ∨ S) ∨ Z=0					
BGE	$addr16	3	PC ← PC+3+jdisp8 if P/V ∨ S=0					
BLT	$addr16	3	PC ← PC+3+jdisp8 if P/V ∨ S=1					
BLE	$addr16	3	PC ← PC+3+jdisp8 if (P/V ∨ S) ∨ Z=1					
BH	$addr16	3	PC ← PC+3+jdisp8 if Z ∨ CY=0					
BNH	$addr16	3	PC ← PC+3+jdisp8 if Z ∨ CY=1					
BT	saddr.bit, $addr16	3	PC ← PC+3+jdisp8 if (saddr.bit)=1					
	sfr.bit, $addr16	4	PC ← PC+4+jdisp8 if sfr.bit=1					
	A.bit, $addr16	3	PC ← PC+3+jdisp8 if A.bit=1					
	X.bit, $addr16	3	PC ← PC+3+jdisp8 if X.bit=1					
	PSWH.bit, $addr16	3	PC ← PC+3+jdisp8 if PSW_H.bit=1					
	PSWL.bit, $addr16	3	PC ← PC+3+jdisp8 if PSW_L.bit=1					

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BF	saddr.bit, $addr16	4	PC ← PC+4+jdisp8 if (saddr.bit)=0					
	sfr.bit, $addr16	4	PC ← PC+4+jdisp8 if sfr.bit=0					
	A.bit, $addr16	3	PC ← PC+3+jdisp8 if A.bit=0					
	X.bit, $addr16	3	PC ← PC+3+jdisp8 if X.bit=0					
	PSWH.bit, $addr16	3	PC ← PC+3+jdisp8 if PSW_H.bit=0					
	PSWL.bit, $addr16	3	PC ← PC+3+jdisp8 if PSW_L.bit=0					
BTCLR	saddr.bit, $addr16	4	PC ← PC+4+jdisp8 if (saddr.bit)=1 then reset (saddr. bit)					
	sfr.bit, $addr16	4	PC ← PC+4+jdisp8 if sfr.bit=1 then reset sfr.bit					
	A.bit, $addr16	3	PC ← PC+3+jdisp8 if A.bit=1 then reset A.bit					
	X.bit, $addr16	3	PC ← PC+3+jdisp8 if X.bit=1 then reset X.bit					
	PSWH.bit, $addr16	3	PC ← PC+3+jdisp8 if PSW_H.bit=1 then reset PSW_H.bit					
	PSWL.bit, $addr16	3	PC ← PC+3+jdisp8 if PSW_L.bit=1 then reset PSW_L.bit	x	x	x	x	x
BFSET	saddr.bit, $addr16	4	PC ← PC+4+jdisp8 if (saddr.bit)=0 then set (saddr. bit)					
	sfr.bit, $addr16	4	PC ← PC+4+jdisp8 if sfr.bit=0 then set sfr.bit					
	A.bit, $addr16	3	PC ← PC+3+jdisp8 if A.bit=0 then set A.bit					
	X.bit, $addr16	3	PC ← PC+3+jdisp8 if X.bit=0 then set X.bit					
	PSWH.bit, $addr16	3	PC ← PC+3+jdisp8 if PSW_H.bit=0 then set PSW_H.bit					
	PSWL.bit, $addr16	3	PC ← PC+3+jdisp8 if PSW_L.bit=0 then set PSW_L.bit	x	x	x	x	x
DBNZ	r2, $addr16	2	r2 ← r2-1 then PC ← PC+2+jdisp8 if r2≠0					
	saddr, $addr16	3	(saddr) ← (saddr)-1, then PC ← PC+3+jdisp8 if(saddr)≠0					

(17) Context switching instruction: BRKCS, RETCS, RETCSB

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BRKCS	RBn	2	RBS2-0 ← n, PC_H ← R4, R7 ← PSW_H, R6 ← PSW_L, RSS ← 0, IE ← 0					
RETCS	!addr16	3	PC_H ← R5, PC_L ← R4, R5 ← $addr16_H$, R4 ← $addr16_L$, RSW_H ← R7,PSW_L ← R6	R	R	R	R	R
RETCSB	!addr16	4	PC_H ← R5, PC_L ← R4, R5 ← $addr16_H$, R4 ← $addr16_L$,PSW_H ← R7, PSW_L ← R6	R	R	R	R	R

(18) String manipulation instruction: MOVM, MOVBK, XCHM, XCHBK, CMPME, CMPBKE, CMPMNE, CMPBKNE, CMPMC, CMPBKC, CMPMNC, CMPBKNC

(1/2)

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVM	[DE+], A	2	(DE+) ← A, C ← C-1 End if C=0					
	[DE-], A	2	(DE-) ← A, C ← C-1 End if C=0					
MOVBK	[DE+],[HL+]	2	(DE+) ← (HL+), C ← C-1 End if C=0					
	[DE-],[HL-]	2	(DE-) ← (HL-), C ← C-1 End if C=0					
XCHM	[DE+], A	2	(DE+) ←→A, C ← C-1 End if C=0					
	[DE-], A	2	(DE-) ←→A, C ← C-1 End if C=0					
XCHBK	[DE+], [HL+]	2	(DE+) ←→(HL+), C ← C-1 End if C=0					
	[DE-], [HL-]	2	(DE-) ←→(HL-), C ← C-1 End if C=0					
CMPME	[DE+], A	2	(DE+)-A, C ← C-1 End if C=0 or Z=0	x	x	x	V	x
	[DE-], A	2	(DE-)-A, C ← C-1 End if C=0 or Z=0	x	x	x	V	x
CMPBKE	[DE+], [HL+]	2	(DE+)-(HL+), C ← C-1 End if C=0 or Z=0	x	x	x	V	x
	[DE-], [HL-]	2	(DE-)-(HL-), C ← C-1 End if C=0 or Z=0	x	x	x	V	x
CMPMNE	[DE+], A	2	(DE+)-A, C ← C-1 End if C=0 or Z=1	x	x	x	V	x
	[DE-], A	2	(DE-)-A, C ← C-1 End if C=0 or Z=1	x	x	x	V	x
CMPBKNE	[DE+], [HL+]	2	(DE+)-(HL+), C ← C-1 End if C=0 or Z=1	x	x	x	V	x
	[DE-], [HL-]	2	(DE-)-(HL-), C ← C-1 End if C=0 or Z=1	x	x	x	V	x
CMPMC	[DE+], A	2	(DE+)-A, C ← C-1 End if C=0 or CY=0	x	x	x	V	x
	[DE-], A	2	(DE-)-A, C ← C-1 End if C=0 or CY=0	x	x	x	V	x
CMPBKC	[DE+], [HL+]	2	(DE+)-(HL+), C ← C-1 End if C=0 or CY=0	x	x	x	V	x
	[DE-], [HL-]	2	(DE-)-(HL-), C ← C-1 End if C=0 or CY=0	x	x	x	V	x

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CMPMNC	[DE+], A	2	(DE+)-A, C ← C-1 End if C=0 or CY=1	x	x	x	V	x
	[DE-], A	2	(DE-)-A, C ← C-1 End if C=0 or CY=0	x	x	x	V	x
CMPBKNC	[DE+],[HL+]	2	(DE+)-(HL+), C ← C-1 End if C=0 or CY=1	x	x	x	V	x
	[DE-],[HL-]	2	(DE-)-(HL-), C ← C-1 End if C=0 or CY=1	x	x	x	V	x

(19) CPU control instruction: MOV, SWRS, SEL, NOP, EI, DI

Mnemonic	Operands	Bytes	Operation	Flags
				S Z AC P/V CY
MOV	STBC, #byte	4	STBC ← byte (Note)	
	WDM, #byte	4	WDM ← byte (Note)	
SWRS		1	RSS ← $\overline{RSS}$	
SEL	RBn	2	RBS2-0 ← n, RSS ← 0	
	RBn, ALT	2	RBS2-0 ← n, RSS ← 1	
NOP		1	No Operation	
EI		1	IE ← 1 (Enable Interrupt)	
DI		1	IE ← 0 (Disable Interrupt)	

Note: When the operation code of an STBC or WDM register handling instruction is abnormal, an trap interrupt is generated.

Operation when an exception trap interrupt occurs.

$(SP-1) \leftarrow PSW_H$, $(SP-2) \leftarrow PSW_L$

$(SP-3) \leftarrow (PC-4)_H$, $(SP-4) \leftarrow (PC-4)_L$

$PC_L \leftarrow (003CH)$, $PC_H \leftarrow (003DH)$

$SP \leftarrow SP-4$, $IE \leftarrow 0$

18.2 Operation Codes of Instructions

18.2.1 Symbol explanation of operation codes

r,r1

R_3	R_2	R_1	R_0	reg
0	0	0	0	R0
0	0	0	1	R1
0	0	1	0	R2
0	0	1	1	R3
0	1	0	0	R4
0	1	0	1	R5
0	1	1	0	R6
0	1	1	1	R7
1	0	0	0	R8
1	0	0	1	R9
1	0	1	0	R10
1	0	1	1	R11
1	1	0	0	R12
1	1	0	1	R13
1	1	1	0	R14
1	1	1	1	R15

r1

r

r2

C_0	reg
0	C
1	B

rp

P_2	P_1	P_0	reg-pair
0	0	0	RP0
0	0	1	RP1
0	1	0	RP2
0	1	1	RP3
1	0	0	RP4
1	0	1	RP5
1	1	0	RP6
1	1	1	RP7

rp1

Q_2	Q_1	Q_0	reg-pair
0	0	0	RP0
0	0	1	RP4
0	1	0	RP1
0	1	1	RP5
1	0	0	RP2
1	0	1	RP6
1	1	0	RP3
1	1	1	RP7

rp2

S_1	S_0	reg-pair
0	0	VP
0	1	UP
1	0	DE
1	1	HL

Bn: Immediate data for bit

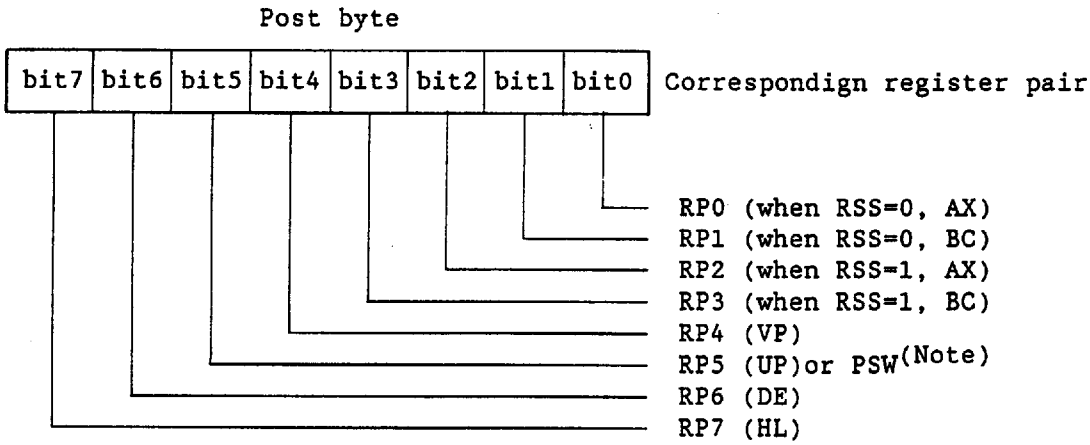
Nn: Immediate data for n

Data: 8-bit immediate data corresponding to byte

Low/High Byte : 16-bit immediate data corresponding to word

Saddr-offset	:	Low-order 8-bit off set data of 16-bit address corresponding to saddr
Sfr-offset	:	Low-order 8-bit data of 16-bit address of special function register (sfr)
Low/High offset	:	8/16-bit offset data in memory addressing in based mode/indexed mode
Low/High Addr.	:	16-bit immediate data corresponding to addr16
jdisp	:	Signed 8-bit two's complement data of relative address distance between top address of next instruction and branch destination address
fa	:	Low-order 11-bits of immediate data corresponding to addr11
ta	:	Low-order five bits of immediate data corresponding to addr5x 1/2
Post Byte	:	8-bit data specifying register pairs for stack handling Each bit is assigned a specific register pair. When a bit is set to 1, its corresponding register pair is specified for stack handling. (See Fig. 18-1.)

Fig. 18-1 8-bit Data Specifying Register Pairs for Stack Handling



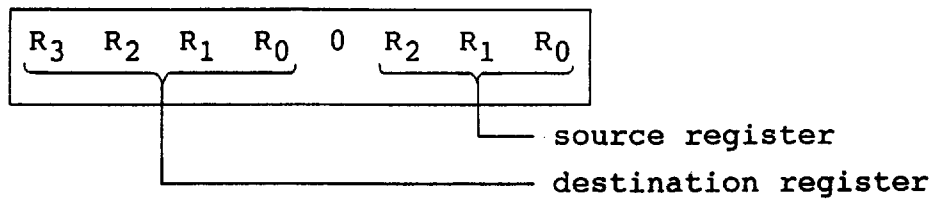
0	Save/restore operation in/from stack memory is not performed
1	Save/restore operation in/from stack memory is performed

Note: RP5 (UP) for PUSH/POP instruction or PSW for PUSHU/POPU instruction.

Caution 1: If both source and destination are both of registers or both saddr and saddrp in the operand field of MOV r, rl, ADD saddr, saddr, etc., the codes are as follows:

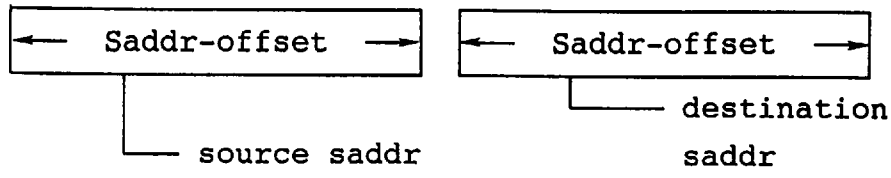
- When both are registers or register pairs, the destinatnion specification code precedes the source specification code.

Example:



- When both are saddr or saddrp, the preceding 1-byte data becomes offset data specifying the source and the following 1-byte data becomes offset data specifying the destination.

Example:



Caution 2: If a special function register (SFR) mapped in FF00H-FF1FH is described in operand sfr or sfrp, short direct addressing rather than SFR addressing is applied and the operation code of the instruction with operand saddr or saddrp is generated.

Example: AND A, P5

Operation code	10011100	00000101

AND A, PM5

Operation code	00000001	10011100	00100101

In this example, since short direct addressing is applied to the AND A, P5 instruction, the operation code becomes shorter than that when SFR addressing is applied.

18.2.2 Operation codes in memory addressing modes

Table 18-5 lists the codes of the mod and mem parts in the operation code field determined corresponding to the contents described in mem in the operand field.

Table 18-5 Codes of mod and mem Parts in Operation Code Field

mod / mem	1 0110	1 0111	0 0110	0 1010
	Register indirect mode	Based indexed mode	Based mode	Indexed mode
0 0 0	[DE+](Note)	[DE+A]	[DE+byte]	word[DE]
0 0 1	[HL+](Note)	[HL+A]	[DE+byte]	word[A]
0 1 0	[DE-](Note)	[DE+A]	[HL+byte]	word[HL]
0 1 1	[HL-](Note)	[HL+B]	[UP+byte]	word[B]
1 0 0	[DE](Note)	[VP+DE]	[VP+byte]	-
1 0 1	[HL](Note)	[VP+HL]	-	-
1 1 0	[VP]	-	-	-
1 1 1	[UP]	-	-	-

Note: If the code is described in mem in the MOV instruction operand field, the MOV instruction becomes a dedicated 1-byte instruction.

Remarks: If the based or indexed mode is described in mem, the 8-bit or 16-bit offset data corresponding to byte or word is added to the third byte and later.

18.2.3 Operation code list

(1) 8-bit data transfer instruction: MOV, XCH

(1/2)

Mnemonic	Operands	Operation code			
		B1	B2	B3	
		B4	B5		
MOV	r1, #byte	1 0 1 1 1 R_2 R_1 R_0	← Data →		
	saddr, #byte	0 0 1 1 1 0 1 0	← Saddr-offset →	← Data →	
	sfr, #byte	0 0 1 0 1 0 1 1	← Sfr-offset →	← Data →	
	r, r1	0 0 1 0 0 1 0 0	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0		
	A, r1	1 1 0 1 0 R_2 R_1 R_0			
	A, saddr	0 0 1 0 0 0 0 0	← Saddr-offset →		
	saddr, A	0 0 1 0 0 0 1 0	← Saddr-offset →		
	saddr, saddr	0 0 1 1 1 0 0 0	← Saddr-offset →	—Saddr-offset→	
	A, sfr	0 0 0 1 0 0 0 0	← Sfr-offset →		
	sfr, A	0 0 0 1 0 0 1 0	← Sfr-offset →		
	A, mem (Note)	0 1 0 1 1 mem			
		0 0 0 mod	0 mem 0 0 0 0	← Low Offset →	
		← High Offset →			
	mem, A (Note)	0 1 0 1 0 mem			
		0 0 0 mod	1 mem 0 0 0 0	← Low Offset →	
		← High Offset →			
	A, [saddrp]	0 0 0 1 1 0 0 0	← Saddr-offset →		
	[saddrp], A	0 0 0 1 1 0 0 1	← Saddr-offset →		
	A, !addr16	0 0 0 0 1 0 0 1	1 1 1 1 0 0 0 0	← Low Addr. →	
		← High Addr. →			
	!addr16, A	0 0 0 0 1 0 0 1	1 1 1 1 0 0 0 1	← Low Addr. →	
		← High Addr. →			
	PSWL, #byte	0 0 1 0 1 0 1 1	1 1 1 1 1 1 1 0	← Data →	
	PSWH, #byte	0 0 1 0 1 0 1 1	1 1 1 1 1 1 1 1	← Data →	
	PSWL, A	0 0 0 1 0 0 1 0	1 1 1 1 1 1 1 0		
	PSWH, A	0 0 0 1 0 0 1 0	1 1 1 1 1 1 1 1		
	A, PSWL	0 0 0 1 0 0 0 0	1 1 1 1 1 1 1 0		
	A, PSWH	0 0 0 1 0 0 0 0	1 1 1 1 1 1 1 1		

Note: If [DE], [HL], [DE+], [DE-], [HL+], or [HL-] is described in mem, the 1-byte code results.

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
XCH	A, rl	1 1 0 1 1 R_2 R_1 R_0		
	r, rl	0 0 1 0 0 1 0 1	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0	
	A, mem	0 0 0 mod	0 mem 0 1 0 0	← Low Offset →
		← High Offset →		
	A, saddr	0 0 1 0 0 0 0 1	← Saddr-offset →	
	A, sfr	0 0 0 0 0 0 0 1	0 0 1 0 0 0 0 1	← Sfr-offset →
	A, [saddr]	0 0 1 0 0 0 1 1	← Saddr-offset →	
	saddr, saddr	0 0 1 1 1 0 0 1	← Saddr-offset →	←Saddr-offset →

(2) 16-bit data transfer instruction: MOVW, XCHW

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
MOVW	rpl, #word	0 1 1 0 0 Q_2 Q_1 Q_0	← Low Byte →	← High Byte →
	saddrp, #word	0 0 0 0 1 1 0 0	← Saddr-offset →	← Low Byte →
		← High Byte →		
	sfrp, #word	0 0 0 0 1 0 1 1	← Sfr-offset →	← Low Byte →
		← High Byte →		
	rp, rpl	0 0 1 0 0 1 0 0	P_3 P_2 P_0 0 1 Q_2 Q_1 Q_0	
	AX, saddrp	0 0 0 1 1 1 0 0	← Saddr-offset →	
	saddrp, AX	0 0 0 1 1 0 1 0	← Saddr-offset →	
	saddrp, saddrp	0 0 1 1 1 1 0 0	← Saddr-offset →	←Saddr-offset →
	AX, sfrp	0 0 0 1 0 0 0 1	← Sfr-offset →	
	sfrp, AX	0 0 0 1 0 0 1 1	← Sfr-offset →	
	rpl, !addr16	0 0 0 0 1 0 0 1	1 0 0 0 0 Q_2 Q_1 Q_0	← Low Addr. →
		← High Addr. →		
	!addr16, rpl	0 0 0 0 1 0 0 1	1 0 0 0 0 Q_2 Q_1 Q_0	← Low Addr. →
		← High Addr. →		
	AX, mem	0 0 0 mod	0 mem 0 0 0 1	← Low-offset →
		← High-offset →		
	mem, AX	0 0 0 mod	1 mem 0 0 0 1	← Low-offset →
		← High-offset →		
XCHW	AX, saddrp	0 0 0 1 1 0 1 1	← Saddr-offset →	
	AX, sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 0 1 1	← Sfr-offset →
	saddrp, saddrp	0 0 1 0 1 0 1 0	← Saddr-offset →	← Saddr-offset→
	rp, rpl	0 0 1 0 0 1 0 1	P_2 P_1 P_0 0 1 Q_2 Q_1 Q_0	
	AX, mem	0 0 0 mod	0 mem 0 1 0 1	← Low-offset →
		← High-offset →		

(3) 8-bit arithmetic and logical instruction: ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP

(1/4)

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
ADD	A, #byte	1 0 1 0 1 0 0 0	← Data →	
	saddr, #byte	0 1 1 0 1 0 0 0	← Saddr-offset →	← Data →
	sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 0 0	← Sfr-offset →
		← Data →		
	r, rl	1 0 0 0 1 0 0 0	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0	
	A, saddr	1 0 0 1 1 0 0 0	← Saddr-offset →	
	A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 0 0	← Sfr-offset →
	saddr, saddr	0 1 1 1 1 0 0 0	← Saddr-offset →	← Saddr-offset →
	A, mem	0 0 0 mod	0 mem 1 0 0 0	← Low Offset →
		← High Offset →		
	mem, A	0 0 0 mod	1 mem 1 0 0 0	← Low Offset →
		← High Offset →		
ADDC	A, #byte	1 0 1 0 1 0 0 1	← Data →	
	saddr, #byte	0 1 1 0 1 0 0 1	← Saddr-offset →	← Data →
	sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 0 1	← Sfr-offset →
		← Data →		
	r, rl	1 0 0 0 1 0 0 1	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0	
	A, saddr	1 0 0 1 1 0 0 1	← Saddr-offset →	
	A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 0 1	← Sfr-offset →
	saddr, saddr	0 1 1 1 1 0 0 1	← Saddr-offset →	← Saddr-offset →
	A, mem	0 0 0 mod	0 mem 1 0 0 1	← Low Offset →
		← High Offset →		
	mem, A	0 0 0 mod	1 mem 1 0 0 1	← Low Offset →
		← High Offset →		

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
SUB	A, #byte	1 0 1 0 1 0 1 0	← Data →	
	saddr, #byte	0 1 1 0 1 0 1 0	← Saddr-offset →	← Data →
	sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 1 0	← Sfr-offset →
		← Data →		
	r, rl	1 0 0 0 1 0 1 0	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0	
	A, saddr	1 0 0 1 1 0 1 0	← Saddr-offset →	
	A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 1 0	← Sfr-offset →
	saddr, saddr	0 1 1 1 1 0 1 0	← Saddr-offset →	← Saddr-offset→
	A, mem	0 0 0 mod	0 mem 1 0 1 0	← Low Offset →
		← High Offset →		
	mem, A	0 0 0 mod	1 mem 1 0 1 0	← Low Offset →
		← High Offset →		
SUBC	A, #byte	1 0 1 0 1 0 1 1	← Data →	
	saddr, #byte	0 1 1 0 1 0 1 1	← Saddr-offset →	← Data →
	sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 0 1 1	← Sfr-offset →
		← Data →		
	r, rl	1 0 0 0 1 0 1 1	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0	
	A, saddr	1 0 0 1 1 0 1 1	← Saddr-offset →	
	A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 0 1 1	← Sfr-offset →
	saddr, saddr	0 1 1 1 1 0 1 1	← Saddr-offset →	← Saddr-offset→
	A, mem	0 0 0 mod	0 mem 1 0 1 1	← Low Offset →
		← High Offset →		
	mem, A	0 0 0 mod	1 mem 1 0 1 1	← Low Offset →
		← High Offset →		
AND	A, #byte	1 0 1 0 1 1 0 0	← Data →	
	saddr, #byte	0 1 1 0 1 1 0 0	← Saddr-offset →	← Data →
	sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 0 0	← Sfr-offset →
		← Data →		
	r, rl	1 0 0 0 1 1 0 0	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0	
	A, saddr	1 0 0 1 1 1 0 0	← Saddr-offset →	

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
AND	A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 0 0	← Sfr-offset →
	saddr, saddr	0 1 1 1 1 1 0 0	← Saddr-offset →	← Saddr-offset→
	A, mem	0 0 0 mod	0 mem 1 1 0 0	← Low Offset →
		← High Offset →		
	mem, A	0 0 0 mod	1 mem 1 1 0 0	← Low Offset →
		← High Offset →		
OR	A, #byte	1 0 1 0 1 1 1 0	← Data →	
	saddr, #byte	0 1 1 0 1 1 1 0	← Saddr-offset →	← Data →
	sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 1 0	← Sfr-offset →
		← Data →		
	r, rl	1 0 0 0 1 1 1 0	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0	
	A, saddr	1 0 0 1 1 1 1 0	← Saddr-offset →	
	A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 1 0	← Sfr-offset →
	saddr, saddr	0 1 1 1 1 1 1 0	← Saddr-offset →	← Saddr-offset→
	A, mem	0 0 0 mod	0 mem 1 1 1 0	← Low Offset →
		← High Offset →		
	mem, A	0 0 0 mod	1 mem 1 1 1 0	← Low Offset →
		← High Offset →		
XOR	A, #byte	1 0 1 0 1 1 0 1	← Data →	
	saddr, #byte	0 1 1 0 1 1 0 1	← Saddr-offset →	← Data →
	sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 0 1	← Sfr-offset →
		← Data →		
	r, rl	1 0 0 0 1 1 0 1	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0	
	A, saddr	1 0 0 1 1 1 0 1	← Saddr-offset →	
	A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 0 1	← Sfr-offset →
	saddr, saddr	0 1 1 1 1 1 0 1	← Saddr-offset →	← Saddr-offset→
	A, mem	0 0 0 mod	0 mem 1 1 0 1	← Low Offset →
		← High Offset →		
	mem, A	0 0 0 mod	1 mem 1 1 0 1	← Low Offset →
		← High Offset →		

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
CMP	A, #byte	1 0 1 0 1 1 1 1	← Data →	
	saddr, #byte	0 1 1 0 1 1 1 1	← Saddr-offset →	← Data →
	sfr, #byte	0 0 0 0 0 0 0 1	0 1 1 0 1 1 1 1	← Sfr-offset →
		← Data →		
	r, rl	1 0 0 0 1 1 1 1	R_3 R_2 R_1 R_0 0 R_2 R_1 R_0	
	A, saddr	1 0 0 1 1 1 1 1	← Saddr-offset →	
	A, sfr	0 0 0 0 0 0 0 1	1 0 0 1 1 1 1 1	← Sfr-offset →
	saddr, saddr	0 1 1 1 1 1 1 1	← Saddr-offset →	← Saddr-offset→
	A, mem	0 0 0 mod	0 mem 1 1 1 1	← Low Offset →
		← High Offset →		
	mem, A	0 0 0 mod	1 mem 1 1 1 1	← Low Offset →
		← High Offset →		

(4) 16-bit arithmatic and logical instruction: ADDW, SUBW, CMPW

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
ADDW	AX, #word	0 0 1 0 1 1 0 1	← Low Byte →	← High Byte →
	saddrp, #word	0 0 0 0 1 1 0 1	← Saddr-offset →	← Low Byte →
		← High Offset →		
	sfrp, #word	0 0 0 0 0 0 0 1	0 0 0 0 1 1 0 1	← Sfr-offset →
		← Low Byte →	← High Byte →	
	rp, rpl	1 0 0 0 1 0 0 0	P_2 P_1 P_0 0 1 Q_2 Q_1 Q_0	
	AX, saddrp	0 0 0 1 1 1 0 1	← Saddr-offset →	
	AX, sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 1 0 1	← Sfr-offset →
	saddrp, saddrp	0 0 1 1 1 1 0 1	← Saddr-offset →	← Saddr-offset→
SUBW	AX, #word	0 0 1 0 1 1 1 0	← Low Byte →	← High Byte →
	saddrp, #word	0 0 0 0 1 1 1 0	← Saddr-offset →	← Low Byte →
		← High Byte →		
	sfrp, #word	0 0 0 0 0 0 0 1	0 0 0 0 1 1 1 0	← Sfr-offset →
		← Low Byte →	← High Byte →	
	rp, rpl	1 0 0 0 1 0 1 0	P_2 P_1 P_0 0 1 Q_2 Q_1 Q_0	
	AX, saddrp	0 0 0 1 1 1 1 0	← Saddr-offset →	← Saddr-offset→
	AX, sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 1 1 0	← Sfr-offset →
	saddrp, saddrp	0 0 1 1 1 1 1 0	← Saddr-offset →	← Saddr-offset→
CMPW	AX, #word	0 0 1 0 1 1 1 1	← Low Byte →	← High Byte →
	saddrp, #word	0 0 0 0 1 1 1 1	← Saddr-offset →	← Low Byte →
		← High Byte →		
	sfrp, #word	0 0 0 0 0 0 0 1	0 0 0 0 1 1 1 1	← Sfr-offset →
		← Low Byte →	← High Byte →	
	rp, rpl	1 0 0 0 1 1 1 1	P_2 P_1 P_0 0 1 Q_2 Q_1 Q_0	
	AX, saddrp	0 0 0 1 1 1 1 1	← Saddr-offset →	
	AX, sfrp	0 0 0 0 0 0 0 1	0 0 0 1 1 1 1 1	← Sfr-offset →
	saddrp, saddrp	0 0 1 1 1 1 1 1	← Saddr-offset →	← Saddr-offset →

(5) Multiplication and division instruction: MULU, DIVUW, MULUW, DIVUX

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
MULU	rl	0 0 0 0 0 1 0 1	0 0 0 0 1 R_2 R_1 R_0	
DIVUW	rl		0 0 0 1 1 R_2 R_1 R_0	
MULUW	rpl		0 0 1 0 1 Q_2 Q_1 Q_0	
DIVUX	rpl		1 1 1 0 1 Q_2 Q_1 Q_0	

(6) Signed multiplication: MULW

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
MULW	rpl		0 0 1 1 1 Q_2 Q_1 Q_0	

(7) Increment and decrement instruction: INC, DEC, INCW, DECW

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
INC	rl	1 1 0 0 0 R_2 R_1 R_0		
	saddr	0 0 1 0 0 1 1 0	← Saddr-offset →	
DEC	rl	1 1 0 0 0 R_2 R_1 R_0		
	saddr	0 0 1 0 0 1 1 1	← Saddr-offset →	
INCW	rp2	0 1 0 0 0 1 S_1 S_0		
	saddr	0 0 0 0 0 1 1 1	1 1 1 0 1 0 0 0	← Saddr-offset →
DECW	rp2	0 1 0 0 1 1 S_1 S_0		
	saddr	0 0 0 0 0 1 1 1	1 1 1 0 1 0 0 1	← Saddr-offset →

(8) Shift and rotate instruction: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
ROR	r1, n	0 0 1 1 0 0 0 0	0 1 N_2 N_1 N_0 R_2 R_1 R_0	
ROL	r1, n	0 0 0 1	0 1 N_2 N_1 N_0 R_2 R_1 R_0	
RORC	r1, n	0 0 0 0	0 0 N_2 N_1 N_0 R_2 R_1 R_0	
ROLC	r1, n	0 0 0 1	0 0 N_2 N_1 N_0 R_2 R_1 R_0	
SHR	r1, n	0 0 0 0	1 0 N_2 N_1 N_0 R_2 R_1 R_0	
SHL	r1, n	0 0 0 1	1 0 N_2 N_1 N_0 R_2 R_1 R_0	
SHRW	rp1, n	0 0 0 0	1 1 N_2 N_1 N_0 Q_2 Q_1 Q_0	
SHLW	rp1, n	0 0 0 1	1 1 N_2 N_1 N_0 Q_2 Q_1 Q_0	
ROR4	[rp1]	0 0 0 0 0 1 0 1	1 0 0 0 1 Q_2 Q_1 Q_0	
ROL4	[rp1]	0 0 0 0 0 1 0 1	1 0 0 1 1 Q_2 Q_1 Q_0	

(9) BCD adjustment instruction: ADJBA, ADJBS

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
ADJBA		0 0 0 0 0 1 0 1	1 1 1 1 1 1 1 0	
ADJBS		0 0 0 0 0 1 0 1	1 1 1 1 1 1 1 1	

(10) Data conversion instruction: CVTBW

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
CVTBW		0 0 0 0 0 1 0 0		

(11) Bit manipulation instruction: MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1

(1/3)

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
MOV1	CY, saddr.bit	0 0 0 0 1 0 0 0	0 0 0 0 0 B_2 B_1 B_0	← Saddr-offset→
	CY, sfr.bit	1 0 0 0	1 B_2 B_1 B_0	← Sfr-offset →
	CY, A.bit	0 0 1 1	1 B_2 B_1 B_0	
	CY, X.bit	0 0 1 1	0 B_2 B_1 B_0	
	CY, PSWH.bit	0 0 1 0	1 B_2 B_1 B_0	
	CY, PSWL.bit	0 0 1 0	0 B_2 B_1 B_0	
	saddr.bit, CY	1 0 0 0	0 0 0 1 0 B_2 B_1 B_0	← Saddr-offset→
	sfr.bit, CY	1 0 0 0	1 B_2 B_1 B_0	← Sfr-offset →
	A.bit, CY	0 0 1 1	1 B_2 B_1 B_0	
	X.bit, CY	0 0 1 1	0 B_2 B_1 B_0	
	PSWH.bit, CY	0 0 1 0	1 B_2 B_1 B_0	
	PSWL.bit, CY	0 0 1 0	0 B_2 B_1 B_0	
AND1	CY, saddr.bit	0 0 0 0 1 0 0 0	0 0 1 0 0 B_2 B_1 B_0	← Saddr-offset→
	CY,/saddr.bit		0 0 1 1 0 B_2 B_1 B_0	← Saddr-offset→
	CY, sfr.bit		0 0 1 0 1 B_2 B_1 B_0	← Sfr-offset →
	CY,/sfr.bit		0 0 1 1 1 B_2 B_1 B_0	← Sfr-offset →
	CY, A.bit	0 0 1 1	0 0 1 0 1 B_2 B_1 B_0	
	CY,/A.bit		0 0 1 1 1 B_2 B_1 B_0	
	CY, X.bit		0 0 1 0 0 B_2 B_1 B_0	
	CY./X.bit		0 0 1 1 0 B_2 B_1 B_0	
	CY, PSWH.bit	0 0 1 0	0 0 1 0 1 B_2 B_1 B_0	
	CY,/PSWH.bit		0 0 1 1 1 B_2 B_1 B_0	
	CY, PSWL.bit		0 0 1 0 0 B_2 B_1 B_0	
	CY,/PSWL.bit		0 0 1 1 0 B_2 B_1 B_0	

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
OR1	CY, saddr.bit	0 0 0 0 1 0 0 0	0 1 0 0 0 B_2 B_1 B_0	← Saddr-offset →
	CY,/saddr.bit		0 1 0 1 0 B_2 B_1 B_0	← Saddr-offset →
	CY, sfr.bit		0 1 0 0 1 B_2 B_1 B_0	← Sfr-offset →
	CY,/sfr,bit		0 1 0 1 1 B_2 B_1 B_0	← Sfr-offset →
	CY, A.bit	0 0 1 1	0 1 0 0 1 B_2 B_1 B_0	
	CY,/A.bit		0 1 0 1 1 B_2 B_1 B_0	
	CY, X.bit		0 1 0 0 0 B_2 B_1 B_0	
	CY,/X.bit		0 1 0 1 0 B_2 B_1 B_0	
	CY, PSWH.bit	0 0 1 0	0 1 0 0 1 B_2 B_1 B_0	
	CY,/PSWH.bit		0 1 0 1 1 B_2 B_1 B_0	
	CY,/PSWL.bit		0 1 0 0 0 B_2 B_1 B_0	
	CY,/PSWL.bit		0 1 0 1 0 B_2 B_1 B_0	
XOR1	CY, saddr.bit	0 0 0 0 1 0 0 0	0 1 1 0 0 B_2 B_1 B_0	← Saddr-offset →
	CY, sfr.bit	1 0 0 0	1 B_2 B_1 B_0	← Sfr-offset →
	CY, A.bit	0 0 1 1	1 B_2 B_1 B_0	
	CY, X.bit	0 0 1 1	0 B_2 B_1 B_0	
	CY, PSWH.bit	0 0 1 0	1 B_2 B_1 B_0	
	CY, PSWL.bit	0 0 1 0	0 B_2 B_1 B_0	
SET1	saddr.bit	1 0 1 1 0 B_2 B_1 B_0	← Saddr-offset →	
	sfr.bit	0 0 0 0 1 0 0 0	1 0 0 0 1 B_2 B_1 B_0	← Sfr-offset →
	A.bit	0 0 1 1	1 B_2 B_1 B_0	
	X.bit	0 0 1 1	0 B_2 B_1 B_0	
	PSWH.bit	0 0 1 0	1 B_2 B_1 B_0	
	PSWL.bit	0 0 1 0	0 B_2 B_1 B_0	
	CY	0 1 0 0 0 0 0 1		
CLR1	saddr.bit	1 0 1 0 0 B_2 B_1 B_0	← Saddr-offset →	
	sfr.bit	0 0 0 0 1 0 0 0	1 0 0 1 1 B_2 B_1 B_0	← Sfr-offset →
	A.bit	0 0 1 1	1 B_2 B_1 B_0	
	X.bit	0 0 1 1	0 B_2 B_1 B_0	
	PSWH.bit	0 0 1 0	0 B_2 B_1 B_0	
	PSWL.bit	0 0 1 0	0 B_2 B_1 B_0	
	CY	0 1 0 0 0 0 0 0		

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
NOT1	saddr.bit	0 0 0 0 1 0 0 0	0 1 1 1 0 B_2 B_1 B_0	← Saddr-offset →
	sfr.bit	1 0 0 0	1 B_2 B_1 B_0	← Sfr-offset →
	A.bit	0 0 1 1	1 B_2 B_1 B_0	
	X.bit	0 0 1 1	0 B_2 B_1 B_0	
	PSWH.bit	0 0 1 0	1 B_2 B_1 B_0	
	PSWL.bit	0 0 1 0	0 B_2 B_1 B_0	
	CY	0 1 0 0 0 0 1 0		

(12) Call and return instruction: CALL, CALLF, CALLT, BRK, RET, RETB, RETI

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
CALL	!addr16	0 0 1 0 1 0 0 0	← Low Addr. →	← High Addr. →
	rp1	0 0 0 0 0 1 0 1	0 1 0 1 1 Q_2 Q_1 Q_0	
	[rp1]	0 0 0 0 0 1 0 1	0 1 1 1 1 Q_2 Q_1 Q_0	
CALLF	!addr1	1 0 0 1 0 ←	fa →	
CALLT	[addr5]	1 1 1 ← ta →		
BRK		0 1 0 1 1 1 1 0		
RET		0 1 0 1 0 1 1 0		
RETB		0 1 0 1 1 1 1 1		
RETI		0 1 0 1 0 1 1 1		

(13) Stack handling instruction: PUSH, PUSHU, POP, POPU, MOVW, INCW, DECW

Mnemonic	Operands	Operation code															
		B1								B2							B3
		B4								B5							
PUSH	sfrp	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	1 ← Sfr-offset →
	post	0	0	1	1	0	1	0	1	← Post Byte →							
	PSW	0	1	0	0	1	0	0	1								
PUSHU	post	0	0	1	1	0	1	1	1	← Post Byte →							
POP	sfrp	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	1 ← Sfr-offset →
	post	0	0	1	1	0	1	0	0	← Post Byte →							
	PSW	0	1	0	0	1	0	0	0								
POPU	post	0	0	1	1	0	1	1	0	← Post Byte →							
MOVW	SP, #word	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0	0 ← Low Byte →
		← High Byte →															
	SP, AX	0	0	0	1	0	0	1	1	1	1	1	1	1	1	0	0
	AX, SP	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0	0
INCW	SP	0	0	0	0	0	1	0	1	1	1	0	0	1	0	0	0
DECW	SP	0	0	0	0	0	1	0	1	1	1	0	0	1	0	0	1

(14) Special instruction: CHKL, CHKLA

Mnemonic	Operands	Operation code															
		B1								B2							B3
		B4								B5							
CHKL	sfr	0	0	0	0	0	1	1	1	1	1	0	0	1	0	0	0 ← Sfr-offset →
CHKLA	sfr	0	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1 ← Sfr-offset →

(15) Unconditional branch instruction: BR

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
BR	!addr16	0 0 1 0 1 1 0 0	← Low Addr. →	← High Addr. →
	rpl	0 0 0 0 0 1 0 1	0 1 0 0 1 Q_2 Q_1 Q_0	
	[rpl]	0 0 0 0 0 1 0 1	0 1 1 0 1 Q_2 Q_1 Q_0	
	$addr16	0 0 0 1 0 1 0 0	← jdisp →	

(16) Conditional branch instruction: BC, BL, BNC, BNL, BZ, BE, BNZ, BNE, BV, BPE, BNV, BPO, BN, BP, BGT, BGE, BLT, BLE, BH, BNH, BT, BF, BRCLR, BFSET, DBNZ

(1/2)

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
BC BL	$ addr16	1 0 0 0 0 0 1 1	← jdisp →	
BNC BNL	$ addr16	1 0 0 0 0 0 1 0	← jdisp →	
BZ BE	$ addr16	1 0 0 0 0 0 0 1	← jdisp →	
BNZ BNE	$ addr16	1 0 0 0 0 0 0 0	← jdisp →	
BV BPE	$ addr16	1 0 0 0 0 1 0 1	← jdisp →	
BNV BPO	$ addr16	1 0 0 0 0 1 0 0	← jdisp →	
BN	$ addr16	1 0 0 0 0 1 1 1	← jdisp →	
BP	$ addr16	1 0 0 0 0 1 1 0	← jdisp →	
BGT	$ addr16	0 0 0 0 0 1 1 1	1 1 1 1 1 0 1 1	← jdisp →
BGE	$ addr16	0 0 0 0 0 1 1 1	1 1 1 1 1 0 0 1	← jdisp →
BLT	$ addr16	0 0 0 0 0 1 1 1	1 1 1 1 1 0 0 0	← jdisp →
BLE	$ addr16	0 0 0 0 0 1 1 1	1 1 1 1 1 0 1 0	← jdisp →
BH	$ addr16	0 0 0 0 0 1 1 1	1 1 1 1 1 1 0 1	← jdisp →
BNH	$ addr16	0 0 0 0 0 1 1 1	1 1 1 1 1 1 0 0	← jdisp →
BT	saddr.bit, $addr16	0 1 1 1 0 B_2 B_1 B_0	← Saddr-offset →	← jdisp →
	sfr.bit, $addr16	0 0 0 0 1 0 0 0	1 0 1 1 1 B_2 B_1 B_0	← Sfr-offset →
		← jdisp →		
	A.bit, $addr16	0 0 0 0 0 0 1 1	1 0 1 1 1 B_2 B_1 B_0	← jdisp →
	X.bit, $addr16	0 0 0 0 0 0 1 1	1 0 1 1 0 B_2 B_1 B_0	← jdisp →
	PSWH.bit,$addr16	0 0 0 0 0 0 1 0	1 0 1 1 1 B_2 B_1 B_0	← jdisp →
	PSWH.bit,$addr16	0 0 0 0 0 0 1 0	1 0 1 1 0 B_2 B_1 B_0	← jdisp →
BF	saddr.bit, $addr16	0 0 0 0 1 0 0 0	1 0 1 0 0 B_2 B_1 B_0	← Saddr-offset →
		← jdisp →		

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
BF	sfr.bit, $addr16	0 0 0 0 1 0 0 0	1 0 1 0 1 B_2 B_1 B_0	← Sfr-offset →
		← jdisp →		
	A.bit, $addr16	0 0 0 0 0 0 1 1	1 0 1 0 1 B_2 B_1 B_0	← jdisp →
	X.bit, $addr16	0 0 0 0 0 0 1 1	1 0 1 0 0 B_2 B_1 B_0	← jdisp →
	PSWH.bit,$addr16	0 0 0 0 0 0 1 0	1 0 1 0 1 B_2 B_1 B_0	← jdisp →
	PSWL.bit,$addr16	0 0 0 0 0 0 1 0	1 0 1 0 0 B_2 B_1 B_0	← jdisp →
BTCLR	saddr.bit, $addr16	0 0 0 0 1 0 0 0	1 1 0 1 0 B_2 B_1 B_0	← Saddr-offset →
		← jdisp →		
	sfr.bit, $addr16	0 0 0 0 0 1 0 0	0 1 1 0 1 B_2 B_1 B_0	← Sfr-offset →
		← jdisp →		
	A.bit, $addr16	0 0 0 0 0 0 1 1	1 1 0 1 1 B_2 B_1 B_0	← jdisp →
	X.bit, $addr16	0 0 0 0 0 0 1 1	1 1 0 1 0 B_2 B_1 B_0	← jdisp →
	PSWH.bit,$addr16	0 0 0 0 0 0 1 0	1 1 0 1 1 B_2 B_1 B_0	← jdisp →
	PSWL.bit,$addr16	0 0 0 0 0 0 1 0	1 1 0 1 0 B_2 B_1 B_0	← jdisp →
BFSET	saddr.bit, $addr16	0 0 0 0 1 0 0 0	1 1 0 0 0 B_2 B_1 B_0	← Saddr-offset →
		← jdisp →		
	sfr.bit, $addr16	0 0 0 0 1 0 0 0	1 1 0 0 1 B_2 B_1 B_0	← Sfr-offset →
		← jdisp →		
	A.bit, $addr16	0 0 0 0 0 0 1 1	1 1 0 0 1 B_2 B_1 B_0	← jdisp →
	X.bit, $addr16	0 0 0 0 0 0 1 1	1 1 0 0 0 B_2 B_1 B_0	← jdisp →
	PSWH.bit,$addr16	0 0 0 0 0 0 1 0	1 1 0 0 1 B_2 B_1 B_0	← jdisp →
	PSWL.bit,$addr16	0 0 0 0 0 0 1 0	1 1 0 0 0 B_2 B_1 B_0	← jdisp →
DBNZ	r2, $addr16	0 0 1 1 0 0 1 C_0	← jdisp →	
	saddr, $addr16	0 0 1 1 1 0 1 1	← Saddr-offset →	← jdisp →

(17) Context switching instruction: BRKCS, RETCS, RETCSB

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
BRKCS	RBn	0 0 0 0 0 1 0 1	1 1 0 1 1 N_2 N_1 N_0	
RETCS	!addr16	0 0 1 0 1 0 0 1	← Low Addr. →	← High-Addr. →
RETCSB	!addr16	0 0 0 0 1 0 0 1	1 1 1 0 0 0 0 0	← Low-Addr. →
		← High Add. →		

(18) String manipulation instruction: MOVM, MOVBK, XCHM, XCHBK, CMPME, CMPBKE, CMPMNE, CMPBKNE, CMPMC, CMPBKC, CMPMNC, CMPBKNC

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
MOVM	[DE+], A	0 0 0 1 0 1 0 1	0 0 0 0 0 0 0 0	
	[DE-], A	0 0 0 1 0 1 0 1	0 0 0 1 0 0 0 0	
MOVBK	[DE+], [HL+]	0 0 0 1 0 1 0 1	0 0 1 0 0 0 0 0	
	[DE-], [HL-]	0 0 0 1 0 1 0 1	0 0 1 1 0 0 0 0	
XCHM	[DE+], A	0 0 0 1 0 1 0 1	0 0 0 0 0 0 0 1	
	[DE-], A	0 0 0 1 0 1 0 1	0 0 0 1 0 0 0 1	
XCHBK	[DE+], [HL+]	0 0 0 1 0 1 0 1	0 0 1 0 0 0 0 1	
	[DE-], [HL-]	0 0 0 1 0 1 0 1	0 0 1 1 0 0 0 1	
CMPME	[DE+], A	0 0 0 1 0 1 0 1	0 0 0 0 0 1 0 0	
	[DE-], A	0 0 0 1 0 1 0 1	0 0 0 1 0 1 0 0	
CMPBKE	[DE+], [HL+]	0 0 0 1 0 1 0 1	0 0 1 0 0 1 0 0	
	[DE-], [HL-]	0 0 0 1 0 1 0 1	0 0 1 1 0 1 0 0	
CMPMNE	[DE+], A	0 0 0 1 0 1 0 1	0 0 0 0 0 1 0 1	
	[DE-], A	0 0 0 1 0 1 0 1	0 0 0 1 0 1 0 1	
CMPBKNE	[DE+], [HL+]	0 0 0 1 0 1 0 1	0 0 1 0 0 1 0 1	
	[DE-], [HL-]	0 0 0 1 0 1 0 1	0 0 1 1 0 1 0 1	
CMPMC	[DE+], A	0 0 0 1 0 1 0 1	0 0 0 0 0 1 1 1	
	[DE-], A	0 0 0 1 0 1 0 1	0 0 0 1 0 1 1 1	
CMPBKC	[DE+], [HL+]	0 0 0 1 0 1 0 1	0 0 1 0 0 1 1 1	
	[DE-], [HL-]	0 0 0 1 0 1 0 1	0 0 1 1 0 1 1 1	
CMPMNC	[DE+], [HL+]	0 0 0 1 0 1 0 1	0 0 0 0 0 1 1 0	
	[DE-], [HL-]	0 0 0 1 0 1 0 1	0 0 0 1 0 1 1 0	
CMPBKNC	[DE+], [HL+]	0 0 0 1 0 1 0 1	0 0 1 0 0 1 1 0	
	[DE-], [HL-]	0 0 0 1 0 1 0 1	0 0 1 1 0 1 1 0	

(19) CPU control instruction: MOV, SWRS, SEL, NOP, EI, DI

Mnemonic	Operands	Operation code		
		B1	B2	B3
		B4	B5	
MOV	STBC, #byte	0 0 0 0 1 0 0 1	1 1 0 0 0 0 0 0	← Data →
		← Data →		
	WDM, #byte	0 0 0 0 1 0 0 1	1 1 0 0 0 0 0 0	← Data →
		← Data →		
SWRS		0 1 0 0 0 0 1 1		
SEL	RBn	0 0 0 0 0 1 0 1	1 0 1 0 1 N_2 N_1 N_0	
	RBn, ALT	0 0 0 0 0 0 0 0	1 0 1 1 1 N_2 N_1 N_0	
NOP		0 0 0 0 0 0 0 0		
EI		0 1 0 0 1 0 1 1		
DI		0 1 0 0 1 0 1 0		

18.3 Instruction Clocks

18.3.1 Explanation of column of clocks

(1) Conditions to calculate number of execution clocks

The conditions to calculate the number of instruction execution clocks under the column of Clocks are as follows:

(a) Sufficient operation codes are always entered in an instruction queue, and when EXU requires an operation code, the operation code can be immediately read.
(b) The stack pointer points to main RAM (FE00H-FEFFH).
(c) Addresses indicated by using mem, !addr16, [saddrp], [DE+], [DE-], [HL+], [HL-], and [rpl] point to main RAM (FE00H-FEFFH).
(d) Only the number of microprogram execution clocks at EXU is counted (the time required from clearing the instruction queue to reading the operation code at the branch destination when a branch is taken during execution of an instruction such as BR, CALL, RET, BRK, or RETI or interrupt service is not contained).

The number of instruction execution clocks is a value calculated by assuming these conditions. Thus, the actual number of clocks when a program is executed may be greater than that listed under the column of Clocks. The reason why it is greater is described below:

(a) When operation code is read from instruction queue
When EXU reads an operation code, if the instruc-

tion queue does not contain any operation code, EXU waits until an operation code is entered in the instruction queue. Particularly, if branch processing occurs, the instruction queue becomes empty from once clearing the instruction queue to reading the operation code at the branch destination, EXU always enters the wait state.

(b) When data in memory other than main RAM is referenced

① When data is read

EXU waits from BCU starting a bus cycle to completing data read.

② When data is written

If EXU issues a data write request to BCU, it can immediately execute the next instruction. However, since BCU cannot acknowledge another processing request occurring from EXU during data write processing execution, EXU enters the wait state (in which it cannot perform memory reference other than main RAM, SFR reference, or branch processing) until BCU terminates write processing.

Particularly, when the instruction queue does not contain any, operation code fetch takes precedence over in write processing into external memory or peripheral RAM, thus when a write bus cycle is to be started cannot be specified. Therefore, when EXU enters the wait state cannot be specified due to timing contention with write processing.

③ Contention between memory reference other than main RAM or branch processing and operation code fetch

When EXU issues a request for memory reference other than main RAM or for branch processing to BCU, if BCU executes a bus cycle of operation code fetch, the memory reference or branch

processing request is not acknowledged until BCU terminates the operation code fetch bus cycle; EXU enters the wait state.

(2) Classification of column of clocks

The number of instruction clocks varies depending on the memory area accessed or to which a branch is taken by the instruction.

- Internal ROM: At internal ROM fetch
- IRAM: When internal dual port RAM (0FE00H-0FEFFH) is accessed
- PRAM: When internal RAM area other than IRAM is accessed
- SFR: When special function register is accessed
- EMEM: When external memory is accessed

(3) n under column of Clocks

- Shift and rotate instructions: Number of shift bits.
- Stack handling instruction: Number of saved/restored registers.
- String manipulation instruction: Number of times a given instruction is executed until the condition is satisfied and an exit is made from loop.

(4) "/" under column of Clocks

- "/": a/b means a or b

18.3.2 Clock list

(1) 8-bit data transfer instruction: MOV, XCH

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
MOV	r1, #byte	2	—	2	—	—	—
	saddr, #byte	3		3		6	
	sfr(Note), #byte	3		—			6
	r, r1	2		3		—	—
	A, r1	1		2			
	A, saddr	2		3		6	
	saddr, A	2					
	saddr, saddr	3		4		10	
	A, sfr	2		—		6	6
	sfr, A	2					
	A, mem	1-4	(See detail for table 18-6 1/8, 2/8)				
	mem, A	1-4					
	A, [saddrp]	2	—	6	—	9	9
	[saddrp], A	2		4		7	7
	A, !addr16,	4	6	6	6	6	6
	!addr16, A	4	—	5	5	5	5
	PSWL, #byte	3		—	—	6	—
	PSWH, #byte	3					
	PSWL, A	2					
	PSWH, A	2					
	A, PSWL	2					
	A, PSWH	2					
XCH	A, r1	1	—	4	—	—	—
	r, r1	2					
	A, mem	2-4	(See detail for table 18-6 3/8)				
	A, saddr	2	—	5	—	11	—
	A, sfr	3		—		13	13
	A, [saddrp]	2		7		10	—
	saddr, saddr	3		8		20	

Note: If STBC or WDM is described in sfr, the instruction becomes another dedicated instruction and the number of bytes and the number of clocks differ from those listed here.

(2) 16-bit data transfer instruction: MOVW, XCHW

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
MOVW	rpl, #word	3	—	3	—	—	—
	saddrp, #word	4		4		7	
	sfrp, #word	4		—		4	
	rp, rpl	2		3		—	
	AX, saddrp	2				6	
	saddrp, AX	2					
	saddrp, saddrp	2		4		10	
	AX, sfrp	2		—		6	
	sfrp, AX	2					
	rpl, !addr16	4	7	7	7	7	7
	!addr16, rpl	4	—	5	5	5	5
	AX, mem	2-4	(See detail for table 18-6 4/8, 5/8)				
	mem, AX	2-4					
XCHW	AX, saddrp	2	—	5	—	11	—
	AX, sfrp	3		—		13	
	saddrp, saddrp	3		8		20	
	rp, rpl	2		4		—	
	AX, mem	2-4	(See detail for table 18-6 6/8)				

(3) 8-bit arithmetic and logical instruction: ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP

(1/2)

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
ADD	A, #byte	2	—	2	—	—	—
	saddr, #byte	3		4		10	
	sfr, #byte	4		—		12	12
	r, rl	2		3		—	—
	A, saddr	2		4		7	
	A, sfr	3		—		9	9
	saddr, saddr	3		5		14	—
	A, mem	2-4	(See detail for table 18-6 7/8, 8/8)				
	mem, A	2-4					
ADDC	A, #byte	2	—	2	—	—	—
	saddr, #byte	3		4		10	
	sfr, #byte	4		—		12	12
	r, rl	2		3		—	—
	A, saddr	2		4		7	
	A, sfr	3		—		9	9
	saddr, saddr	3		5		14	—
	A, mem	2-4	(See detail for table 18-6 7/8, 8/8)				
	mem, A	2-4					
SUB	A, #byte	2	—	2	—	—	—
	saddr, #byte	3		4		10	
	sfr, #byte	4		—		12	12
	r, rl	2		3		—	—
	A, saddr	2		4		7	
	A, sfr	3		—		9	9
	saddr, saddr	3		5		14	—
	A, mem	2-4	(See detail for table 18-6 7/8, 8/8)				
	mem, A	2-4					
SUBC	A, #byte	2	—	2	—	—	—
	saddr, #byte	3		4		10	
	sfr, #byte	4		—		12	12
	r, rl	2		3		—	—
	A, saddr	2		4		7	
	A, sfr	3		—		9	9
	saddr, saddr	3		5		14	—
	A, mem	2-4	(See detail for table 18-6 7/8, 8/8)				
	mem, A	2-4					

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
AND	A, #byte	2	—	2	—	—	—
	saddr, #byte	3		4		10	
	sfr, #byte	4		—		12	12
	r, rl	2		3		—	—
	A, saddr	2		4		7	
	A, sfr	3		—		9	9
	saddr, saddr	3		5		14	—
	A, mem	2-4	(See detail for table 18-6 7/8, 8/8)				
	mem, A	2-4					
OR	A, #byte	2	—	2	—	—	—
	saddr, #byte	3		4		10	
	sfr, #byte	4		—		12	12
	r, rl	2		3		—	—
	A, saddr	2		4		7	
	A, sfr	3		—		9	9
	saddr, saddr	3		5		14	—
	A, mem	2-4	(See detail for table 18-6 7/8, 8/8)				
	mem, A	2-4					
XOR	A, #byte	2	—	2	—	—	—
	saddr, #byte	3		4		10	
	sfr, #byte	4		—		12	12
	r, rl	2		3		—	—
	A, saddr	2		4		7	
	A, sfr	3		—		9	9
	saddr, saddr	3		5		14	—
	A, mem	2-4	(See detail for table 18-6 7/8, 8/8)				
	mem, A	2-4					
CMP	A, #byte	2	—	2	—	—	—
	saddr, #byte	3		4		7	
	sfr, #byte	4		—		9	9
	r, rl	2		3		—	—
	A, saddr	2		4		7	
	A, sfr	3		—		9	9
	saddr, saddr	3		5		11	—
	A, mem	2-4	(See detail for table 18-6 7/8, 8/8)				
	mem, A	2-4					

(4) 16-bit arithmetic and logical instruction: ADDW, SUBW, CMPW

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
ADDW	AX, #word	3	—	3	—	—	—
	saddrp, #word	4		5		11	
	sfrp, #word	5		—		13	
	rp, rpl	2		3		—	
	AX, saddrp	2		4		7	
	AX, sfrp	3		—		12	
	saddrp, saddrp	3		5		14	
SUBW	AX, #word	3	—	3	—	—	—
	saddrp, #word	4		5		11	
	sfrp, #word	5		—		13	
	rp, rpl	2		3		—	
	AX, saddrp	2		4		7	
	AX, sfrp	3		—		12	
	saddrp, saddrp	3		5		14	
CMPW	AX, #word	3	—	3	—	—	—
	saddrp, #word	4		5		8	
	sfrp, #word	5		—		10	
	rp, rpl	2		3		—	
	AX, saddrp	2		4		7	
	AX, sfrp	3		—		9	
	saddrp, saddrp	3		5		11	

(5) Multiplication and Division instruction: MULU, DIVUW, MULUW, DIVUX

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
MULU	rl	2	—	14	—	—	—
DIVUW	rl	2	—	23	—	—	—
MULUW	rpl	2	—	22	—	—	—
DIVUX	rpl	2	—	43	—	—	—

(6) Signed multiplication instruction: MULW

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
MULW	rpl	2	(See detail for table 18-6 8/8)				

(7) Increment and decrement instruction: INC, DEC, INCW, DECW

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
INC	rl	1	—	2	—	—	—
	saddr	2		3		9	
DEC	rl	1	—	2	—	—	—
	saddr	2		3		9	
INCW	rp2	1	—	2	—	—	—
	saddrp	3		4		10	
DECW	rp2	1	—	2	—	—	—
	saddrp	3		4		10	

(8) Shift rotate instruction: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
ROR	rl, n	2	—	6+n	—	—	—
ROL	rl, n	2	—	6+n	—	—	—
RORC	rl, n	2	—	6+n	—	—	—
ROLC	rl, n	2	—	6+n	—	—	—
SHR	rl, n	2	—	6+n	—	—	—
SHL	rpl, n	2	—	6+n	—	—	—
SHRW	rpl, n	2	—	6+n	—	—	—
ROR4	[rpl]	2	—	8	—	—	—
ROL4	[rpl]	2	—	8	—	—	—

(9) BCD adjustment instruction: ADJBA, ADJBS

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
ADJBA		2	—	5	—	—	—
ADJBS		2	—	5	—	—	—

(10) Data conversion instruction: CVTBW

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
CVTBW		1	—	3	—	—	—

(11) Bit manipulation instruction: MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1

(1/2)

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
MOV1	CY, saddr.bit	3	—	6	—	9	—
	CY, sfr.bit	3		—			9
	CY, A.bit	2		6		—	—
	CY, X.bit	2					
	CY, PSWH.bit	2		—		6	
	CY, PSWL.bit	2					
	saddr.bit, CY	3		5		11	
	sfr.bit, CY	3		—		8	8
	A.bit, CY	2		7		—	—
	X.bit, CY	2					
	PXWH.bit, CY	2		—		8	
	PSWL.bit, CY	2					
AND1	CY, saddr.bit	3	—	6	—	9	—
	CY, /saddr.bit	3					
	CY, sfr.bit	3		—			9
	CY, /sfr.bit	3					
	CY, A.bit	2		6		—	—
	CY, /A.bit	2					
	CY, /X.bit	2					
	CY, X.bit	2					
	CY, PSWH.bit	2		—		6	
	CY, /PSWH.bit	2					
	CY, PSWL.bit	2					
	CY, /PSWL.bit	2					
OR1	CY, saddr.bit	3	—	6	—	9	—
	CY, /saddr.bit	3					
	CY, sfr.bit	3		—			9
	CY, /sfr.bit	3					
	CY, A.bit	2		6		—	—
	CY, /A.bit	2					
	CY, X.bit	2					
	CY, /X.bit	2					
	CY, PSWH.bit	2		—		6	
	CY,/PSWH.bit	2					
	CY, PSWL.bit	2					
	CY, /PSWL.bit	2					

Mnemonic	Operands	Bytes	Clocks				
			Internal ROM	IRAM	PRAM	SFR	EMEM
XOR1	CY, saddr.bit	3	—	6	—	9	—
	CY, sfr.bit	3		—			9
	CY, A.bit	2		6		—	—
	CY, X.bit	2					
	CY, PSWH.bit	2		—		6	
	CY, PSWL.bit	2					
SET1	saddr.bit, CY	2	—	4	—	10	—
	sfr.bit, CY	3		—		11	11
	A.bit	2		6		—	—
	X.bit	2					
	PXWH.bit	2		—		7	
	PSWL.bit	2					
	CY	1				2	
CLR1	saddr.bit	2	—	4	—	10	—
	sfr.bit	3		—		11	11
	A.bit	2		6		—	—
	X.bit	2					
	PSWH.bit	2		—		7	
	PSWL.bit	2					
	CY	1				2	
NOT1	saddr.bit	3	—	5	—	11	—
	sfr.bit	3		—			11
	A.bit	2		6		—	—
	X.bit	2					
	PSWH.bit	2		—		7	
	PSWL.bit	2					
	CY	1				2	

(12) Call and return instructions: CALL, CALLF, CALLT, BRK, RET, RETB, RETI

Mnemonic	Operands	Bytes	Clocks
CALL	!addr16	3	6
	rpl	2	7
	[rpl]	2	10
CALLF	!addr11	2	6
CALLT	[addr5]	1	15
BRK		1	17
RET		1	6
RETB		1	10
RETI		1	10

(13) Stack handling instructions: PUSH, PUSHU, POP, POPU, MOVW, INCW, DECW

Mnemonic	Operands	Bytes	Clocks
PUSH	sfrp	3	10
	post	2	$3+4c_1+6n$
	PSW	1	3
PUSHU	post	2	$4+4c_1+6n$
POP	sfrp	3	9
	post	2	$3+4c_2+7n$
	PSW	1	5
POPU	post	2	$5+4c_2+7n$
MOVW	SP, #word	4	5
	SP, AX	2	4
	AX, SP	2	4
INCW	SP	2	4
DECW	AX, SP	2	4

Remarks: n, c1, and c2 in PUSH, PUSHU, POP, and POPU post instructions denote the following values:

n: Number of registers described as post
c_1: Number of "0" bits to the left of "1" nearest to LSB within post bit pattern
c_2: Number of "0" bits to the right of "1" nearest to MSB within post bit pattern

Example: post=00010100 (specify RP2 and RP4) C_1=4, C_2=3, n=2
post=00110000 (specify RP4 and RP5) C_1=2, C_2=4, n=2

(14) Special instruction: CHKL, CHKLA

Mnemonic	Operands	Bytes	Clocks
CHKL	sfr	3	12
CHKLA	sfr	3	12

(15) Unconditional branch instruction: BR

Mnemonic	Operands	Bytes	Clocks
BR	!addr16	3	4
	rpl	2	4
	[rpl]	2	8
	$addr16	2	4

(16) Conditional branch instruction: BC, BL, BNC, BNL, BZ, BE, BNZ, BNE, BV, BPE, BNV, BPO, BN, BP, BGT, BGE, BLT, BLE, BH, BNH, BT, BF, BTCLR, BFSET, DBNZ (1/2)

Mnemonic	Operands	Bytes	Clocks
BC	$addr16	2	4
BL			
BNC	$addr16	2	4
BNL			
BZ	$addr16	2	4
BE			
BNZ	$addr16	2	4
BNE			
BV	$addr16	2	4
BPE			
BNV	$addr16	2	4
BPO			
BN	$addr16	2	4
BP	$addr16	2	4
BGT	$addr16	3	5
BGE	$addr16	3	5
BLT	$addr16	3	5
BLE	$addr16	3	5
BH	$addr16	3	5
BNH	$addr16	3	5

Mnemonic	Operands	Bytes	Clocks					
			In-line			Branch		
			IRAM	SFR	EMEM	IRAM	SFR	EMEM
BT	saddr.bit, $addr16	3	7	10	—	7	10	—
	sfr.bit, $addr16	4	—	8	8	—	8	8
	A.bit, $addr16	3	8	—	—	8	—	—
	X.bit, $addr16	3						
	PSWH.bit, $addr16	3	—	8		—	8	
	PSWL.bit, $addr16	3						
BF	saddr.bit, $addr16	4	7	10	—	7	10	—
	sfr.bit, $addr16	4	—	8	8	—	8	8
	A.bit, $addr16	3	8	—	—	8	—	—
	X.bit, $addr16	3						
	PSWH.bit, $addr16	3	—	8		—	8	
	PSWL.bit, $addr16	3						
BTCLR	saddr.bit, $addr16	4	10	16	—	8	14	—
	sfr.bit, $addr16	4	—	10	10	—	8	8
	A.bit, $addr16	3	10	—	—	8	—	—
	X.bit, $addr16	3						
	PSWH.bit, $addr16	3	—	10		—	8	
	PSWL.bit, $addr16	3						
BFSET	saddr.bit, $addr16	4	10	16	—	8	14	—
	sfr.bit, $addr16	4	—	10	10	—	8	8
	A.bit, $addr16	3	10	—	—	8	—	—
	X.bit, $addr16	3						
	PSWH.bit, $addr16	3	—	10		—	8	
	PSWL.bit, $addr16	3						
DBNZ	r2, $addr16	2	6	—	—	5	—	—
	saddr, $addr16	3	7	13		6	12	

(17) Context switching instruction: BRKCS, RETCS, RETCSB

Mnemonic	Operands	Bytes	Clocks
BRKCS	RBn	2	7
RETCS	!addr16	3	5
RETCSB	!addr16	4	5

(18) String instruction: MOVM, MOVBK, XCHM, XCHBK, CMPME, CMPBKE, CMPMNE, CMPBKNE, CMPMC, CMPBKC, CMPMNC, CMPBKNC

Mnemonic	Operands	Bytes	Clocks
MOVM	[DE+], A	2	3+6n
	[DE-], A	2	3+6n
MOVBK	[DE+], [HL+]	2	3+10n
	[DE-], [HL-]	2	3+10n
XCHM	[DE+], A	2	3+10n
	[DE-], A	2	3+10n
XCHBK	[DE+], [HL+]	2	3+16n
	[DE-], [HL-]	2	3+16n
CMPME	[DE+], A	2	3+10n
	[DE-], A	2	3+10n
CMPBKE	[DE+], [HL+]	2	3+13n
	[DE-], [HL-]	2	3+13n
CMPMNE	[DE+], A	2	3+10n
	[DE-], A	2	3+10n
CMPBKNE	[DE+], [HL+]	2	3+13n
	[DE-], [HL-]	2	3+13n
CMPMC	[DE+], A	2	3+10n
	[DE-], A	2	3+10n
CMPBKC	[DE+], [HL+]	2	3+13n
	[DE-], [HL-]	2	3+13n
CMPMNC	[DE+], A	2	3+10n
	[DE-], A	2	3+10n
CMPBKNC	[DE+], [HL+]	2	3+13n
	[DE-], [HL-]	2	3+13n

(19) CPU control instruction: MOV, SWRS, SEL, NOP, EI, DI

Mnemonic	Operands	Bytes	Clocks
MOV	STBC, #byte	4	11
	WDM, #byte	4	11
SWRS		1	2
SEL	RBn	2	3
	RBn, ALT	2	3
NOP		1	2
EI		1	3
DI		1	3

Table 18-6 Instruction Execution Cycle List (1/8)

Instruction group	Mnemonic	Operands	Bytes	Clocks				
				Internal ROM	IRAM	PRAM	SFR	EMEM
8-bit data transfer	MOV	A, [DE] A, [HL] A, [DE+] A, [HL+] A, [DE-] A, [HL-]	1	7+n	6	7+n	7+n	7+n
		A, [VP] A, [UP]	2	12+n	10	12+n	12+n	12+n
		A, [DE+A] A, [HL+A] A, [DE+B] A, [HL+B] A, [VP+DE] A, [VP+HL]	2	10+n	8	10+n	10+n	10+n
		A, [DE+byte] A, [HL+byte] A, [VP+byte] A, [UP+byte] A, [SP+byte]	3	10+n	8	10+n	10+n	10+n
		A, word[A] A, word[B] A, word[DE] A, word[HL]	4	11+n	9	11+n	11+n	11+n

Remarks: n is the number of wait states specified in the PWC register.

Table 18-6 Instruction Execution Cycle List (2/8)

Instruction group	Mnemonic	Operands	Bytes	Clocks				
				Internal ROM	IRAM	PRAM	SFR	EMEM
8-bit data transfer	MOV	[DE], A [HL], A [DE+], A [HL+], A [DE-], A [HL-], A	1	-	4	4	4	4
		[VP], A [UP], A	2	-	8	8	8	8
		[DE+A], A [HL+A], A [DE+B], A [HL+B], A [VP+DE], A [VP+HL], A	2	-	6	6	6	6
		[DE+byte], A [HL+byte], A [VP+byte], A [UP+byte], A [SP+byte], A	3	-	6	6	6	6
		word[A], A word[B], A word[DE], A word[HL], A	4	-	7	7	7	7

Table 18-6 Instruction Execution Cycle List (3/8)

Instruction group	Mnemonic	Operands	Bytes	Clocks				
				Internal ROM	IRAM	PRAM	SFR	EMEM
8-bit data transfer	XCH	A, [DE] A, [HL] A, [DE+] A, [HL+] A, [DE-] A, [HL-] A, [VP] A, [UP]	2	-	11	13+n	13+n	13+n
		A, [DE+A] A, [HL+A] A, [DE+B] A, [HL+B] A, [VP+DE] A, [VP+HL]	2	-	9	11+n	11+n	11+n
		A, [DE+byte] A, [HL+byte] A, [VP+byte] A, [UP+byte] A, [SP+byte]	3	-	9	11+n	11+n	11+n
		A, word[A] A, word[B] A, word[DE] A, word[HL]	4	-	10	12+n	12+n	12+n

Remarks: n is the number of wait states specified in the PWC register.

Table 18-6 Instruction Execution Cycle List (4/8)

Instruction group	Mnemonic	Operands	Bbytes	Clocks				
				Internal ROM	IRAM	PRAM	SFR	EMEM
16-bit data transfer	MOVW	AX, [DE] AX, [HL] AX, [DE+] AX, [HL+] AX, [DE-] AX, [HL-] AX, [VP] AX, [UP]	2	15+2n	10	15+2n	15+2n	15+2n
		AX, [DE+A] AX, [HL+A] AX, [DE+B] AX, [HL+B] AX, [VP+DE] AX, [VP+HL]	2	13+2n	8	13+2n	13+2n	13+2n
		AX, [DE+byte] AX, [HL+byte] AX, [VP+byte] AX, [UP+byte] AX, [SP+byte]	3	13+2n	8	13+2n	13+2n	13+2n
		AX, word[A] AX, word[B] AX, word[DE] AX, word[HL]	4	14+2n	9	14+2n	14+2n	14+2n

Remarks: n is the number of wait states specified in the PWC register.

Table 18-6 Instruction Execution Cycle List (5/8)

Instruction group	Mnemonic	Operands	Bytes	Clocks				
				Internal ROM	IRAM	PRAM	SFR	EMEM
16-bit data transfer	MOVW	[DE], AX [HL], AX [DE+], AX [HL+], AX [DE-], AX [HL-], AX [VP], AX [UP], AX	2	-	8	8	8	8
		[DE+A], AX [HL+A], AX [DE+B], AX [HL+B], AX [VP+DE], AX [VP+HL], AX	2	-	6	6	6	6
		[DE+byte], AX [HL+byte], AX [VP+byte], AX [UP+byte], AX [SP+byte], AX	3	-	6	6	6	6
		word[A], AX word[B], AX word[DE], AX word[HL], AX	4	-	7	7	7	7

Table 18-6 Instruction Execution Cycle List (6/8)

Instruction group	Mnemonic	Operands	bytes	Clocks				
				Internal ROM	IRAM	PRAM	SFR	EMEM
16-bit data transfer	XCHW	AX, [DE] AX, [HL] AX, [DE+] AX, [HL+] AX, [DE-] AX, [HL-] AX, [VP] AX, [UP]	2	-	11	16+2n	16+2n	16+2n
		AX, [DE+A] AX, [HL+A] AX, [DE+B] AX, [HL+B] AX, [VP+DE] AX, [VP+HL]	2	-	9	14+2n	14+2n	14+2n
		AX, [DE+byte] AX, [HL+byte] AX, [VP+byte] AX, [UP+byte] AX, [SP+byte]	3	-	9	14+2n	14+2n	14+2n
		AX, word[A] AX, word[B] AX, word[DE] AX, word[HL]	4	-	10	15+2n	15+2n	15+2n

Remarks: n is the number of wait states specified in the PWC register.

Table 18-6 Instruction Execution Cycle List (7/8)

Instruction group	Mnemonic	Operands	Bytes	Clocks				
				Internal ROM	IRAM	PRAM	SFR	EMEM
8-bit arithmetic and logical	ADD ADDC SUB SUBC AND OR XOR CMP	A, [DE] A, [HL] A, [DE+] A, [HL+] A, [DE-] A, [HL-] A, [VP] A, [UP]	2	10+n	8	10+n	10+n	10+n
		A, [DE+A] A, [HL+A] A, [DE+B] A, [HL+B] A, [VP+DE] A, [VP+HL]	2	10+n	8	10+n	10+n	10+n
		A, [DE+byte] A, [HL+byte] A, [VP+byte] A, [UP+byte] A, [SP+byte]	3	10+n	8	10+n	10+n	10+n
		A, word[A] A, word[B] A, word[DE] A, word[HL]	4	11+n	9	11+n	11+n	11+n

Remarks: n is the number of wait states specified in the PWC register.

Table 18-6 Instruction Execution Cycle List (8/8)

Instruction group	Mnemonic	Operands	Bytes	Clocks				
				Internal ROM	IRAM	PRAM	SFR	EMEM
8-bit arithmetic and logical	ADD ADDC SUB SUBC AND OR XOR CMP	[DE], A [HL], A [DE+], A [HL+], A [DE-], A [HL-], A [VP], A [UP], A	2	10+n	8	10+n	10+n	10+n
		[DE+A], A [HL+A], A [DE+B], A [HL+B], A [VP+DE], A [VP+HL], A	2	10+n	8	10+n	10+n	10+n
		[DE+byte], A [HL+byte], A [VP+byte], A [UP+byte], A [SP+byte], A	3	10+n	8	10+n	10+n	10+n
		word[A], A word[B], A word[DE], A word[HL], A	4	11+n	9	11+n	11+n	11+n

Signed multiplication instruction

Mnemonic	Operands	Bytes	Clocks			
			AX(+) rpl(+)	AX(-) rpl(-)	AX(+) rpl(-)	AX(-) rpl(+)
MULW	rpl	2	24	27	28	28

Remarks: n is the number of wait states specified in the PWC register.

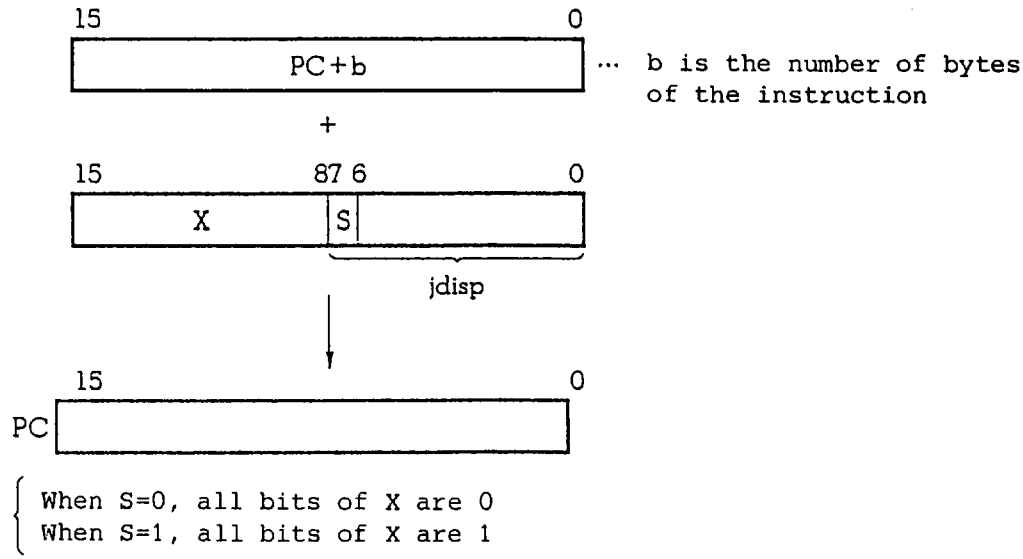
18.4 Instruction Address Addressing

The instruction address is determined by the program counter (PC) contents. Normally, each time one instruction is executed, automatically it is incremented by one for one byte according to the number of bytes of the fetched instruction. When an instruction involving a branch is executed, branch destination address information is set in the PC for a branch according to the addressing modes described below.

18.4.1 Relative addressing

The value resulting from adding 8-bit immediate data (displacement value: jdisp) of operation code to the top address of the following instruction is transferred to the program counter (PC) for a branch. The displacement value is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit. The relative addressing is applied when a BR $addr16 instruction or conditional branch instruction is executed.

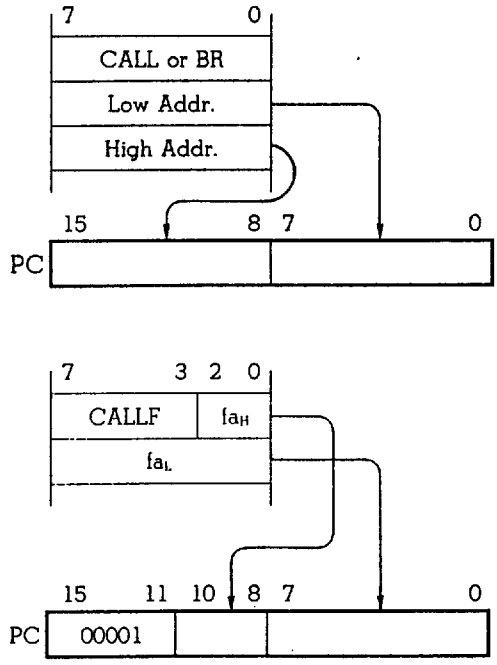
Fig. 18-2 Relative Addressing



18.4.2 Immediate addressing

Immediate data in instruction is transferred to the program counter (PC) for a branch. The immediate addressing is applied when a CALL !addr16, BR !addr16, or CALLF !addr11 instruction is executed. When the CALLF !addr11 instruction is executed, a branch is taken to a fixed area with the high-order 5-bit address determined.

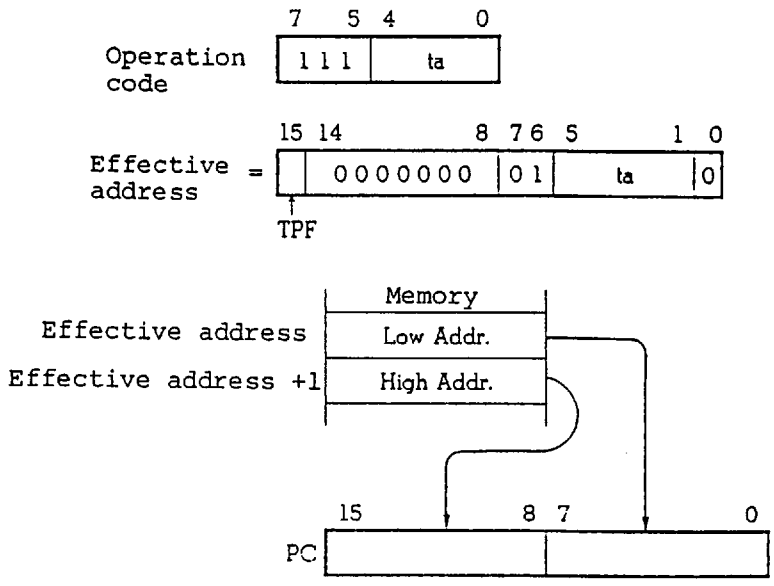
Fig. 18-3 Immediate Addressing



18.4.3 Table indirect addressing

The table contents of a specific location addressed by immediate data of the low-order five bits of operation code (branch destination address) are transferred to the program counter (PC) for a branch. The table indirect addressing is applied when a CALLT [addr5] instruction is executed.

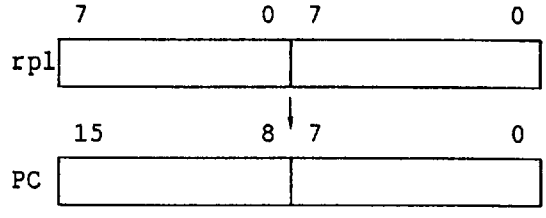
Fig. 18-4 Table Indirect Addressing



18.4.4 Register addressing

The contents of the register pair (RP0-RP7) specified in instruction are transferred to the program counter (PC) for a branch. The register addressing is applied when a BR rpl or CALL rpl instruction is executed.

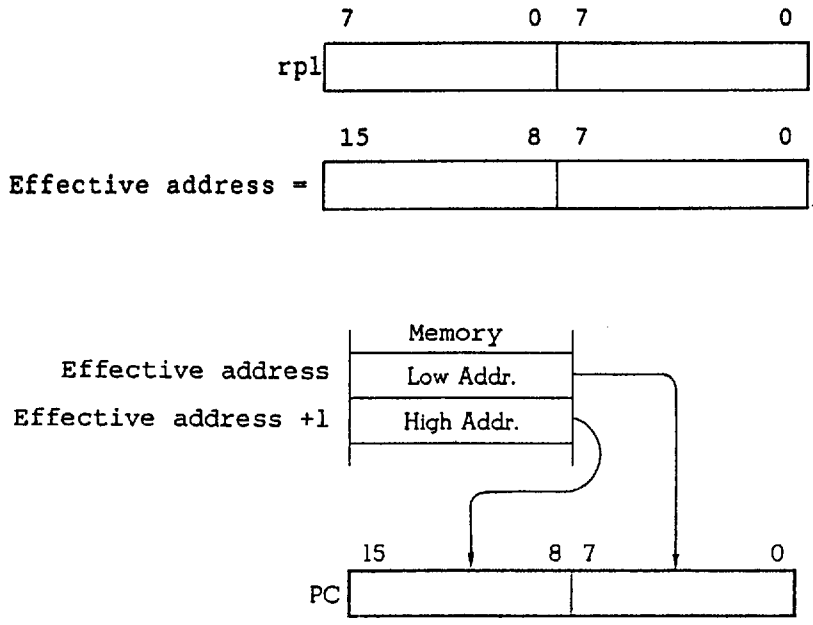
Fig. 18-5 Register Addressing



18.4.5 Register indirect addressing

The 2-byte data in the contiguous locations of memory addressed by the contents of the register pair (RP0-RP7) specified in instruction is transferred to the program counter (PC) for branch. The register indirect addressing is applied when a BR [rpl] or CALL [rpl] instruction is executed.

Fig. 18-6 Register Indirect Addressing



18.5 Operand Address Addressing

This section explains the addressing modes of registers, memory, etc., used as operands in instruction execution.

18.5.1 Register addressing

The general purpose register, specified by the register set selection flag (RSS) and the register specification code (Rn, Pn, or Qn) in instruction, in the register bank specified by the

register bank selection flag (RBSO-RBS2) is accessed as an operand.

The register addressing is applied when an instruction having any of the following operand formats is executed: (When an 8-bit register is specified, one of the eight 8-bit register is specified by setting the three bits of operation code or one of the 16 8-bit registers is specified by setting the four bits of operation code. When a 16-bit register pair is specified, one of the eight register pairs is specified by setting the three bits of operation code.)

Identifier	Description
r	R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11 R12, R13, R14, R15
r1	R0, R1, R2, R3, R4, R5, R6, R7
r2	C, B
rp	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp1	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp2	DE, HL, VP, UP

The function names (X, A, C, B, E, D, L, H, AX, BC, DE, HL, VP, and UP) as well as the absolute names (R0-R15 and RP0-RP7) can be described in r, r1, rp, and rp1. See Tables 9-2 and 9-3 for the correspondence between the absolute and function names.

Example 1: MOV A, r1

Operation code

1	1	0	1	OR_2	R_1	R_0

To select the R2 register as r1, describe the instruction as follows: (The R2 register becomes C register when RSS=0.)

MOV A, R2

The operation code of the instruction is as follows:

Operation code	1	1	0	1	0	0	1	0

Example 2: INCW rp2

Operation code	0	1	0	0	0	1	S_1	S_0

To select the DE register pair as rp2, describe the instruction as follows:

INCW DE

The operation code of the instruction is as follows:

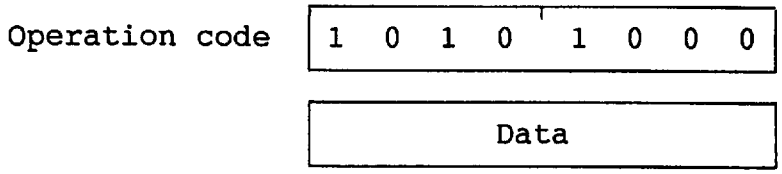
Operation code	0	1	0	0	0	1	1	0

18.5.2 Immediate addressing

8-bit data or 16-bit data as an operand is contained in operation code. The immediate addressing is applied when an instruction having either of the following operands is executed:

Identifier	Description
byte	Label or 8-bit numeric value
word	Label or 16-bit numeric value

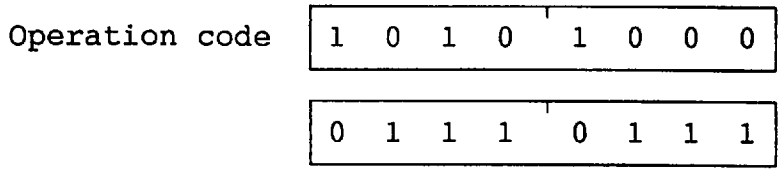
Example: ADD A, #byte



To adopt 77H as byte, describe the instruction as follows:

ADD A, #77H

The operation code of this instruction is as follows:

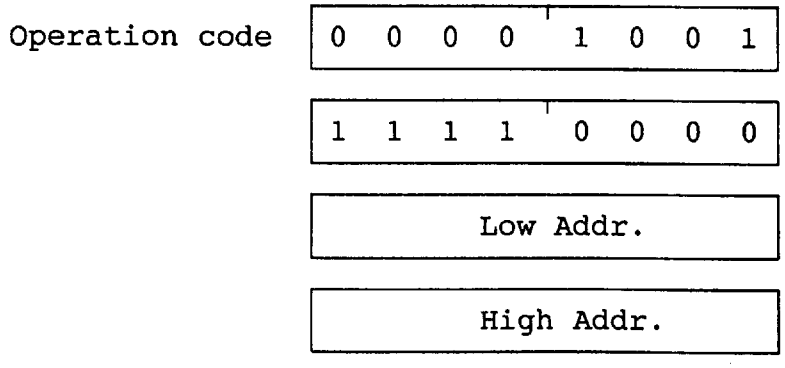


18.5.3 Direct addressing

The memory to be handled is addressed by immediate data in instruction as operand address. The direct addressing is applied when an instruction having the following operand is executed:

Identifier	Description
addr16	Label or 16-bit numeric value

Example: MOV A, !addr16



To adopt FE00H as addr16, describe the instruction as follows:

MOV A, !0FE00H

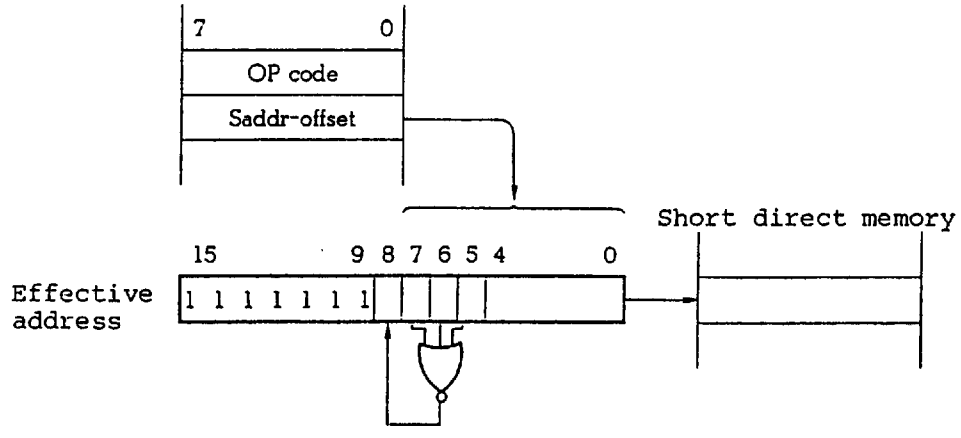
The operation code of this instruction is as follows:

Operation code	0	0	0	0	1	0	0	1
	1	1	1	1	0	0	0	0
	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	0

18.5.4 Short direct addressing

Fixed space memory to be handled is addressed directly by 8-bit immediate addressing in instruction. The short direct addressing is applied to the 256-byte space of addresses FE20H-FF1FH, Internal RAM (short direct memory) is mapped in FE20H-FEFFH and the special function registers (SFRs) are mapped in FF00H-FF1FH. When the 8-bit immediate data is 20H-FFH, bit 8 of the effective address is set to 0; when 00H-1FH, bit 8 is set to 1.

Fig. 18-7 Short Direct Addressing



The short direct addressing is applied when an instruction containing saddr or saddrp in the operand field is executed. When an instruction containing saddrp is executed, the 2-byte data at the memory location addressed by the effective address and the next memory location (data at even-odd addresses where the least significant bit of the effective address is ignored) is accessed.

Identifier	Description
saddr	Label or numeric value in the range of FE20H-FF1FH
saddrp	Label or numeric value in the range of FE20H-FF1EH (even number)

Example: MOV saddr, saddr

Operation code	0	0	1	1	1	0	0	0
	Saddr-offset							
	Saddr-offset							

To adopt FE30H as saddr of the first operand and FE50H as saddr of the second operand, describe the instruction as follows:

MOV 0FE20H, 0FE50H

The operation code of this instruction is as follows:

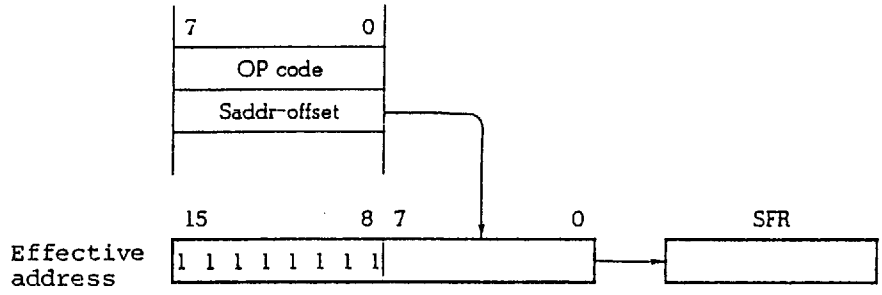
Operation code	0	0	1	1	1	0	0	0
	0	0	1	0	0	0	0	0
	0	1	1	0	0	0	0	0

18.5.5 Special function register (SFR) addressing

Special functions register (SFR) mapped in memory is addressed by 8-bit immediate data in instruction.
The SFR-mapped space to which the special function register addressing is applied is the 256-byte space of addresses FF00H-FFFFH. However, the SFRs mapped in FF00H-FF1FH are accessed by using not only the SFR addressing but also the short direct addressing .

Remarks: In the assembler package manufactured by NEC (RA78K/III), short direct addressing is automatically (forcibly) used for instructions for SFRs mapped in FF00H-FF1FH.

Fig. 18-8 Special Function Register Addressing



Identifier	Description
sfr	Special function register symbol
sfrp	Special function register symbol of special function register that can be handled in 16-bits units.

Example: MOV sfr, A

Operation code	0	0	0	1	0	0	1	0
	Saddr-offset							

To specify PM0 as sfr, describe the instruction as follows:

Example: MOV PM0, A

The operation code of this instruction is as follows:

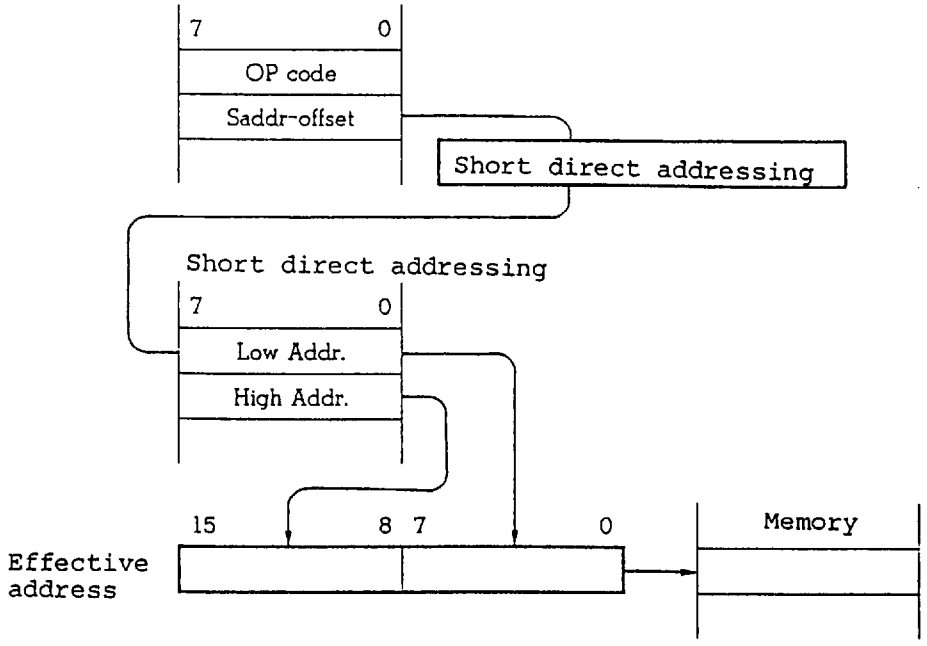
Operation code	0	0	0	1	0	0	1	0
	0	0	1	0	0	0	0	0

18.5.6 Short direct memory indirect addressing

The memory to be handled is addressed by the 2-byte contents of the continuous short direct memory locations addressed by 8-bit immediate data in instruction.

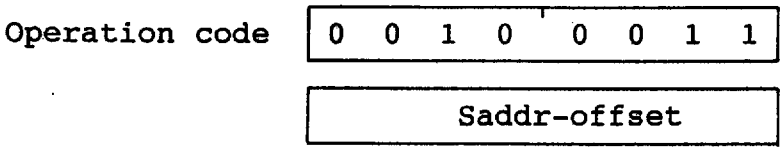
The short direct memory indirect addressing is applied when an instruction having [saddrp] in the operand field is executed.

Fig. 18-9 Short Direct Memory Indirect Addressing



Identifier	Description
[saddrp]	[label or numeric value in the range of FE20H-FF1FH (even value)]

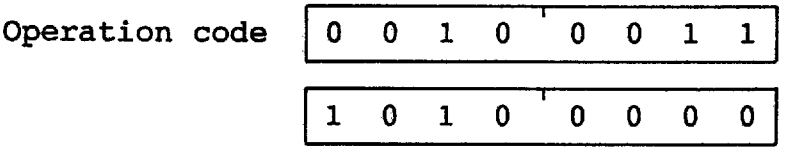
Example: XCH A, [saddrp]



To adopt FEA0H as saddrp, describe the instruction as follows:

```
XCH A, [0FA0H]
```

The operation code of this instruction is as follows:



18.5.7 Register indirect addressing

The memory to be handled is addressed by the contents of the register pair, specified by the register set selection flag (FSS) and the register pair specification code in instruction, in the register bank specified by the register bank selection flag (RBS1-RBS2).
The register indirect addressing is applied when an instruction having any of the following operand formats is executed:

Identifier	Description
mem	[DE], [HL], [DE+], [HL+], [DE-], [HL-], [VP], [UP]
[rpl]	[RP0], [RP1], [RP2], [RP3], [RP4], [RP5], [RP6], [RP7]

The register indirect addressing using register pair DE or HL provides the function which increments or decrements the register pair contents by one for the next addressing.
To use this function, enter [DE+], [HL+], [DE-], or [HL-] in mem in the operand field.

Example 1: MOV A, mem

Operation code:
- When register indirect mode [DE], [HL], [DE+], [HL+], [DE-], or [HL-] is described in mem

0	1	0	1	1	mem

- When any other register indirect mode than the above is described

0	0	0	1	0	1	1	0

0	mem	0	0	0	0

To specify [DE] in mem, describe the instruction as follows:

MOV A, [DE]

The operation code of this instruction is as follows:

Operation code	0	1	0	1	1	1	0	0

Example 2: ROR4 [rpl]

Operation code	0	0	0	0	0	1	0	1
	1	0	0	0	1	Q_2	Q_1	Q_0

To select RP0 as rpl, describe the instruction as follows:

ROR4 [RP0]

The operation code of this instruction is as follows:

Operation code	0	0	0	0	0	1	0	1
	1	0	0	0	1	0	0	0

Example 3: ADD A, mem

Operation code (when register indirect mode is described)

Operation code	0	0	0	1	0	1	1	0
	0	mem			1	0	0	0

To specify [HL+] in mem, describe the instruction as follows:

ADD A, [HL+]

The operation code of this instruction is as follows:

Operation code

0	0	0	1	0	1	1	0

0	0	0	1	1	0	0	0

18.5.8 Based addressing

The memory to be handled is addressed by the sum of the contents of the 16-bit register or register pair (DE, SP, HL, U or VP), specified by the register set selection flag (RSS) and the addressing code (mem) in instruction, in the register bank specified by the register bank selection flag (RBS0-RBS2) and the 8-bit immediate data given in the operand.

The based addressing is applied when an instruction having the following operand format is executed:

Identifier	Description
mem	[DE+byte], [SP+byte], [HL+byte], [UP+byte], [VP+byte]

Example: AND A, mem

Operation code

0	0	0	0	0	1	1	0

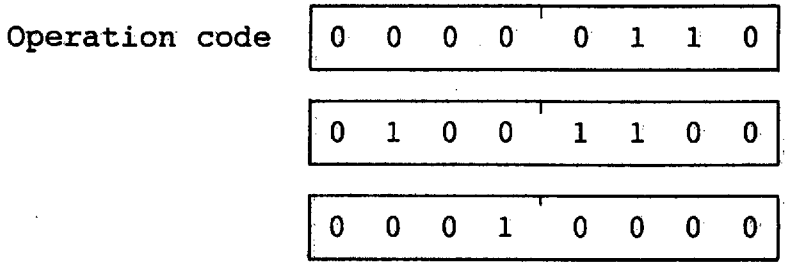
0	mem	1	1	0	0

offset

To select the based addressing with the sum of the register pair VP contents and 10H as mem, describe the instruction as follows:

```
AND A, [VP+10H]
```

The operation code of this instruction is as follows:

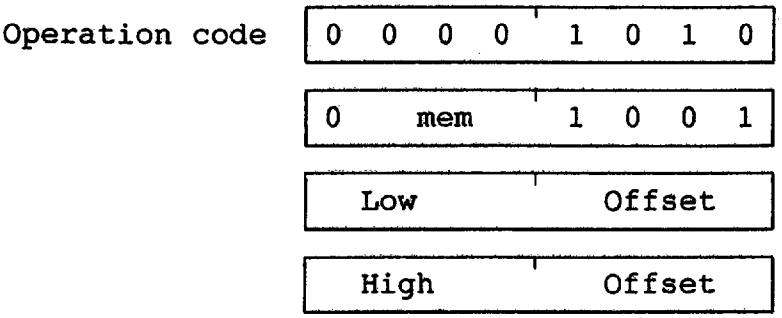


18.5.9 Indexed addressing

The memory to be handled is addressed by the sum of the contents of the 8-bit register or 16-bit register pair (A, B, DE, or HL), specified by register set selection flag (RSS) and the addressing code (mem) in instruction, in the register bank specified by the register bank selection flag (RBS0-RBS2) and the 16-bit immediate data given in the operand.
The indexed addressing is applied when an instruction having the following operand format is executed:

Identifier	Description
mem	word[DE], word[A], word[HL], word[B]

Example: ADDC A, mem



To select the indexed addressing with the sum of the register pair DE contents and 4010H as mem, describe the instruction as follows:

ADDC A, 4010H[DE]

The operation code of this instruction is as follows:

Operation code	0	0	0	0	1	0	1	0
	0	0	0	0	1	0	0	1
	0	0	0	1	0	0	0	0
	0	1	0	0	0	0	0	0

18.5.10 Based indexed addressing

The memory to be handled is addressed by the sum of the contents of the 16-bit register (DE, HL, or VP) and 8-bit or 16-bit register (A, B, DE or HL), specified by the register set selection flag (RSS) and the addressing code (mem) in instruction, in the register bank specified by the register bank selection flag (RBS0-RBS2).
The based indexed addressing is applied when an instruction having the following operand format is executed:

Identifier	Description
mem	[DE+A], [HL+A], [DE+B], [HL+B], [VP+DE], [VP+HL]

Example: OR A, mem

Operation code	0	0	0	1	0	1	1	1
	0	mem			1	0	0	1

To select the based indexed addressing with the sum of the register pair HL and register B contents as mem, describe the instruction as follows:

SUBC A, [HL+B]

The operation code of this instruction is as follows:

Operation code	0	0	0	1	0	1	1	1
	0	0	1	1	1	0	1	1

18.6 Explanation of Instructions

18.6.1 8-bit data transfer instructions

MOV rl, #byte

Function: rl ← byte byte=00H-FFH
Transfer the 8-bit immediate data specified in the second operand to the 8-bit register specified in the first operand.

Flag operation: No change

Description example: MOV R1, #4DH;Set 4DH in register R1.

MOV saddr, #byte

Function: (saddr) ← byte saddr=FE20H-FF1FH
byte=00H-FFH
Transfer the 8-bit immediate data specified in the second operand to the short direct memory addressed in the first operand.
Describe the short direct memory address or label in the first operand saddr as it is.

Flag operation: No change

Description example: MOV 0FE40H, #40H;Store 40H in address FE40H.

MOV sfr, #byte

Function: sfr ← byte byte=00H-FFH
Transfer the 8-bit immediate data specified in the second operand to the special function register sfr specified in the first operand.

Caution: If STBC or WDM is described as sfr, dedicated operation code different from that of the MOV sfr, #byte instruction is generated. (See 18.6.19)

Flag operation: No change

Description example: MOV PM0, #0H;Specify port 0 as output port.

MOV r, r1

Function: r ← r1
Transfer the contents of the 8-bit register specified in the second operand to the 8-bit register specified in the first operand.

Flag operation: No change

Description example:
```
SEL RB0      ; Select bank 0.
MOV R15, R1; Transfer the R1 (A) register
               contents to R15 (H) register.
```

MOV A, r1

Function: A ← r1
Transfer the contents of the 8-bit register specified in the second operand to the A register.

Flag operation: No change

MOV A, saddr

Function: A ← (saddr) saddr=FE20H-FF1FH
Transfer the contents of the short direct memory addressed in the second operand to the A register.
Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation: No change

Description example:
```
MOV A, 0FE40H; Transfer the contents of address FE40H to the A register of the specified bank.
```

MOV saddr, A

Function: (saddr) ← A saddr=FE20H-FF1FH
Transfer the A register contents to the short direct memory addressed in the first operand.
Describe the short direct memory address or label in the first operand saddr as it is.

Flag operation: No change

Description example:
```
SEL RB2     ; Select bank 2.
MOV R1, R0; Transfer X register contents
            to memory address FE30H.
MOV 0FE30H, A
```

MOV saddr, saddr

Function: (saddr) ← (saddr) saddr=FE20H-FF1FH
Transfer the contents of the short direct memory addressed in the second operand (source) to the short direct memory addressed in the first operand (destination).

Describe the short direct memory address or label in each of the first and second operands saddr as it is.

Flag operation: No change

Description example: MOV 0FE18H, 0FEC2H; Transfer contents of address FEC2H to address FE18H.

MOV A, sfr

Function: A ← sfr
Transfer the contents of the special function register specified in the second operand to the A register.

Flag operation: No change

Description example: MOV A, ADCR; Transfer A/D conversion result to the A register of the current register bank selected.

MOV sfr, A

Function: sfr ← A
Transfer the A register contents to the special function register specified in the first operand.

Flag operation: No change

Description example: MOV PM1, #00H; Specify output port mode for port 1.
MOV P1, A ; Output A register contents from port 1.

MOV A, mem

Function: A ← (mem)

Transfer the contents of the memory addressed by the memory addressing described in the second operand to the A register.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after transferring the data.

Flag operation: No change

Description example:

```
MOVW RP6, #3000H; DE (RP6) ← 3000H
MOV  A, [DE+]   ; Transfer contents of
                  memory address 3000H to
                  A register (increment DE
                  contents by one after
                  transfer).
```

MOV mem, A

Function: (mem) ← A

Transfer the A register contents to the memory addressed by the memory addressing described in the first operand.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after transferring the data.

Flag operation: No change

Description example:

```
        MOV  R2,  #0FFH  ; Set FFH in C (R2) register.
        MOVW RP6, #3000H ; Set 3000H in DE register pair.
        MOV  R1,  #0H    ; Set 0H in A (R1) register.
LOOP:   MOV  [DE+], A    ; Set A register contents in address
                           3000H (increment DE contents by
                           one after transfer).
        DBNZ C, $LOOP    ; Initialize contents of memory
                           addresses 3000H-30F to 0H.
```

MOV A, [saddrp]

Function: A ← ((saddrp)) saddrp=FE20H-FF1EH

Transfer the contents of the memory addressed by the contents of the short direct memory addressed in the second operand to the A register.

Describe the short direct memory address or label in the second operand saddrp as it is (even address only).

Flag operation: No change

MOV [saddrp], A

Function: ((saddrp)) ← A saddrp=FE20H-FF1EH

Transfer the A register contents to the memory addressed by the contents of the short direct memory addressed in the first operand.

Describe the short direct memory address or label in the first operand saddrp as it is (even address only).

Flag operation: No change

MOV A, !addr16

Function: A ← (addr16) addr16=0000H-FFFFH
Transfer the contents of the memory addressed by the 16-bit immediate data specified in the second operand to the A register.

Flag operation: No change

Description example: MOV A, EXAM; Transfer contents of memory addressed by label EXAM to A register.

MOV !addr16, A

Function: (addr16) ← A addr16=0000H-FFFFH
Transfer the A register contents to the memory addressed by the 16-bit immediate data specified in the first operand.

Flag operation: No change

MOV PSWL, #byte

Function: PSW_L ← byte byte=00H-FFH
Transfer the 8-bit immediate data specified in the second operand to the low order eight bits of PSW.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	x	x

MOV PSWH, #byte

Function: PSW_H ← byte byte=00H-FFH
Transfer the 8-bit immediate data specified in the second operand to the high-order eight bits of the PSW.

Flag operation: No change

MOV PSWL, A

Function: PSW_L ← A
Transfer the A register contents to the low-order eight bits of the PSW.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	x	x

MOV PSWH, A

Function: PSW_H ← A
Transfer the A register contents to the high-order eight bits of the PSW.

Flag operation: No change

MOV A, PSWL

Function: A ← PSW_L
Transfer the low-order 8-bit contents of the PSW to the A register.

Flag operation: No change

MOV A, PSWH

Function: A ← PSW_H
Transfer the high-order 8-bit contents of the PSW to the A register.

Flag operation: No change

XCH A, r1

Function: A ↔ r1
Exchange the contents of the A register and the 8-bit register specified in the second operand.

Flag operation: No change

XCH r, r1

Function: r ↔ r1
Exchange the content of the 8-bit registers specified in the first and second operands.

Flag operation: No change

XCH A, mem

Function: A ↔ (mem)
Exchange the contents of the A register and the memory addressed by the memory addressing described in the second operand.
If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after exchanging the data.

Flag operation: No change

Description example:

```
        MOV   R2, #0FH      ; C(R2) ← 10H
        MOVW  RP7, #FE10H   ; HL (RP7) ← FE10H
        MOV   R1, #0H       ; A (R1) ← 00H
LOOP:   XCH   [HL-],A       ; Exchange the contents of A
                              register and address FE10H
                              decrement the HL contents by
                              one after transfer).
        DBNZ  C, $LOOP      ; Shift the FE01H-FE10H contents
                              forward one address (address
                              FE10H=00H).
```

XCH A, saddr

Function: A ⟷ (saddr) saddr=FE20H-FF1FH

Exchange the contents of the A register and the short direct memory addressed in the second operand.

Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation: No change

XCH A, sfr

Function: A ⟷ sfr

Exchange the contents of the A register and the special function register specified in the second operand.

Flag operation: No change

Description example: XCH A, RXB; Exchange the contents of A register and serial receive buffer.

XCH A, [saddrp]

Function: A ⟷ ((saddrp)) saddrp=FE20H-FF1EH

Exchange the contents of the A register and the memory addressed by the contents of the short direct memory addressed in the second operand.

Describe the short direct memory address or label in the second operand saddrp as it is (even address only).

Flag operation: No change

XCH saddr, saddr

Function: (saddr) ⟷ (saddr) saddr=FE20H-FF1FH

Exchange the contents of the short direct memory addressed in the first operand and the short direct memory addressed in the second operand.

Describe the short direct memory address or label in each of the first and second operands saddr as it is.

Flag operation: No change

18.6.2 16-bit data transfer instructions

MOVW rp1, #word

Function: rp1 ← word word=0000H-FFFFH
Transfer the 16-bit immediate data specified in the second operand to the 16-bit register pair specified in the first operand.

Flag operation: No change

Description example: MOVW RP0, #0AA55H; Transfer AA55H to AX register pair.

MOVW saddrp, #word

Function: (saddrp) ← word saddrp=FE20H-FF1EH
word=0000H-FFFFH
Transfer the 16-bit immediate data specified in the second operand to the 2-byte area of the short direct memory addressed in the first operand.
Describe the short direct memory address or label in the first operand saddrp as it is (even address only).

Flag operation: No change

MOVW sfrp, #word

Function: sfrp ← word word=0000H-FFFFH
Transfer the 16-bit immediate data specified in the second operand to the 16-bit special function register specified in the first operand.

Flag operation: No change

Description example: MOVW PWM0, #0FF00H; Set FF00H in PWM0 register.

MOVW rp, rp1

Function: rp ← rp1
Transfer the contents of the 16-bit register pair specified in the second operand to the 16-bit register pair specified in the first operand.

Flag operation: No change

MOVW AX, saddrp

Function: AX ← (saddrp) saddrp=FE20H-FF1EH
Transfer the contents of the 2-byte area of the short direct memory addressed in the second operand to the register pair AX.
Describe the short direct memory address or label in the second operand saddrp as it is (even address only).

Flag operation: No change

Description example: MOVW AX, 0FE30H; Transfer contents of address FE31H and FE30H to AX register pair.

MOVW saddrp, AX

Function: (saddrp) ← AX saddrp=FE20H-FF1EH
Transfer the register pair AX contents to the 2-byte area of the short direct memory addressed in the first operand.
Describe the short direct memory address or label in the first operand saddrp as it is (even address only).

Flag operation: No change

MOVW saddrp, saddrp

Function: (saddrp) ← (saddrp) saddrp=FE20H-FF1EH
Transfer the contents of the 2-byte area of the short direct memory addressed in the second operand to the 2-byte area of the short direct memory addressed in the first operand.
Describe the short direct memory address or label in each of the first and second operands saddrp as it is (even address only).

Flag operation: No change

MOVW AX, sfrp

Function: AX ← sfrp
Transfer the contents of the 16-bit special function register specified in the second operand to the register pair AX.

Flag operation: No change

Description example: MOVW AX, CPT0; Transfer CPT0 register contents to AX register pair.

MOVW sfrp, AX

Function: sfrp ← AX
Transfer the register pair AX contents to the 16-bit special function register specified in the first operand.

Flag operation: No change

MOVW AX, mem

Function: AX ← (mem) mem=0000H-FDFFH (any desired address)
mem=FE00H-FFFFH (limited to even address)

Transfer the contents of the memory addressed by the memory addressing described in the second operand to the register pair AX.
If auto increment ([DE+] or [HL+]) or auto decrement [DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by two after transferring the data.

Flag operation: No change

MOVW mem, AX

Function: (mem) ← AX mem=0000H-FDFFH (any desired address)
mem=FF00H-FFFFH (limited to even address)

Transfer the register pair AX contents to the memory addressed by the memory addressing described in the first operand.
If auto increment ([DE+] or [HL+] or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by two after transferring the data.

Flag operation: No change

MOVW rp1, !addr16

Function: rp1 ← (addr16) addr16=0000H-FDFFH (any desired address)
addr16=FE00H-FFFFH (limited to even addresswes)

Transfer the contents of the 2-byte area of the memory addressed by the 16-bit immediate data specified in the second operand to the 16-bit register pair specified in the first operand.

Flag operation: No change

MOVW !addr16, rpl

Function: (addr16) ← rpl addr16=0000H-FDFFH
(any desired addresses)
addr16=FE00H-FFFFH
(limited to even addresses)

Transfer the contents of the 16-bit register pair specified in the second operand to the 2-byte area of memory addressed by the 16-bit immediate data specified in the first operand.

Flag operation: No change

XCHW AX, saddrp

Function: AX ← (saddrp) saddrp=FE20H-FF1EH

Exchange the contents of the register pair AX and the 2-byte area of the short direct memory addressed in the second operand.

Describe the short direct memory address or label in the second operand saddrp as it is (even address only).

Flag operation: No change

XCHW AX, sfrp

Function: AX ← sfrp

Exchange the contents of the register pair AX and the 16-bit special function register specified in the

second operand.

Flag operation: No change

Description example: Exchange the contents of the register XCHW AX, PWM0L; PWM0 and the register pair AX.

XCHW saddrp, saddrp

Function: (saddrp) ←→ (saddrp) saddrp=FE20H-FF1EH
Exchange the contents of the 2-byte area of the short direct memory addressed in the first operand and the 2-byte area of the short direct memory addressed in the second operand.
Describe the short direct memory address or label in each of the first and second operands saddrp as it is (even address only).

Flag operation: No change

XCHW rp, rp1

Function: rp ←→ rp1
Exchange the contents of the 16-bit register pair specified in the first and second operands.

Flag operation: No change

XCHW AX, mem

Function: AX ←→ (mem) mem=0000H-FDFFH
(any desired addresses)
mem=FE00H-FFFFH
(limited to even addresses)

Exchange the contents of the register pair AX and the 2-byte area of the memory addressed by the memory addressing described in the second operand.
If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by two after transferring the data.

Flag operation: No change

18.6.3 8-bit arithmetic and logical instructions

ADD A, #byte

Function: A, CY ← A+byte byte=00H-FFH

Add the 8-bit immediate data specified in the second operand to the A register contents in binary. Set the carry flag if a carry is generated as a result of the additions. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Description example: ADD A, #40H; Add 40H to A register contents in binary.

ADD saddr, #byte

Function: (saddrp), CY ← (saddr)+byte saddr=FE20H-FF1FH
byte=00H-FFH

Add the 8-bit immediate data specified in the second operand to the contents of the short direct memory addressed in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Describe the short direct memory address or label in the first operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADD sfr, #byte

Function: sfr, CY ← sfr+byte byte=00H-FFH

Add the 8-bit immediate data specified in the second operand to the contents of the special function register specified in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Description example: ADD CR00L #1H; Add 1H to the contents of low-order eight bits of CR00 register in binary and set the result in CR00 register.

ADD r, r1

Function: r, CY ← r+r1

Add the contents of the register specified in the second operand to the contents of the register specified in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADD A, saddr

Function: A, CY ← A+(saddr) saddr=FE20H-FF1FH

Add the contents of the short directly memory addressed in the second operand to the A register contents in binary. Set the carry flag is a carry is generated as a result of the addition.
Reset the carry flag if no carry is generated.
Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADD A, sfr

Function: A, CY ← A+sfr

Add the contents of the special function register specified in the second operand to the A register contents in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADD saddr, saddr

Function: (saddr), CY ← (saddr)+(saddr) saddr=FE20H-FF1FH

Add the contents of the short direct memory addressed in the second operand to the contents of the short direct memory addressed in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Describe the short direct memory address or label in each of the first and second operands saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADD A, mem

Function: A, CY ← A+(mem)

Add the contents of the memory addressed by the memory addressing described in the second operand to the A register contents in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the addition.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADD mem, A

Function: (mem), CY ← (mem)+A

Add the A register contents to the contents of the memory addressed by the memory addressing decribed in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL conents by one after the addition.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Description example:

```
        MOV   R2,  #0FH   ; C ← 0FH
        MOV   RP7, #FE20H; HL ← FE20H
LOOP:   MOV   A,   #10H
        ADD   [HL+80H], A; Add 10H to the FEA0H-FEAFH contents
        DBNZ  C,$LOOP
```

ADDC A, #byte

Function: A, CY ← A+byte+CY byte=00H-FFH

Add the 8-bit immediate data specified in the second operand with the carry flag to the A register contents in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDC saddr, #byte

Function: (saddr), CY ← (saddr)+byte+CY saddr=FE20H-FF1FH
byte=00H-FFH

Add the 8-bit immediate data specified in the second operand with the carry flag to the contents of the short direct memory addressed in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Describe the short direct memory address or label in the first operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Caution: If any of the following special function registers is specified in the first operand as short direct memory, the operation result becomes undefined. Do not describe it in the first operand.

Special function register: P4, P5

ADDC sfr, #byte

Function: sfr, CY ← sfr+byte+CY byte=00H-FFH

Add the 8-bit immediate data specified in the second operand with the carry flag to the contetns of the special function register specified in the first operand in binary. Set the carry flag if a carry is generated as result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Caution: If any of the following special function registers is specified in the first operand, the operation result becomes undefined. Do not describe it in the first operand.

Special function register: P4, P5, PM5, MM, external SFR

ADDC r, r1

Function: r, CY ← r+r1+CY

Add the contents of the 8-bit register specified in the second operand with the carry flag to the contents of the 8-bit register specified in the first operand in binary. Set the carry flag if a carry is

generated as a results of the addition. Set the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDC A, saddr

Function: A, CY ← A+(saddr)+CY saddr=FE20H-FF1FH

Add the contents of the short direct memory addressed in the second operand with the carry flag to the A register contents in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDC A, sfr

Function: A, CY ← A+sfr+CY

Add the contents of the special function register specified in the second operand with the carry flag to the A register contents in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDC saddr, saddr

Function: (saddr), CY ← (saddr)+(saddr)+CY saddr=FE20H-FF1FH
Add the contents of the short direct memory addressed in the second operand with the carry flag to the contents of the short direct memory addressed in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Caution: If any of the following special function registers is specified in the first operand as short direct memory, the operation result becomes undefined. Do not describe it in the first operand.

Special function register: P4, P5

ADDC A, mem

Function: A, CY ← A+(mem)+CY
Add the contents of the memory addressed by the memory addressing described in the second operand with the carry flag to the A registr contents in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.
If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the addition.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDC mem, A

Function: (mem), CY ← (mem)+A+CY

Add the A register contents with the carry flag to the contents of the memory addressed by the memory addressing described in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the addition.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUB A, #byte

Function: A, CY ← A-byte byte=00H-FFH

Subtract the 8-bit immediate data specified in the second operand from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUB saddr, #byte

Function: (saddr), CY ← (saddr)-byte saddr=FE20H-FF1FH
byte=00H-FFH

Subtract the 8-bit immediate data specified in the second operand from the contents of the short direct memory addressed in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Describe the short direct memory address or label in the first operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUB sfr, #byte

Function: sfr, CY ← sfr-byte byte=00H-FFH

Subtract the 8-bit immediate data specified in the second operand from the contents of the special function register specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUB r, r1

Function: r, CY ← r-r1

Subtract the contents of the 8-bit register specified in the second operand from the contents of the 8-bit register specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUB A, saddr

Function: A, CY ← A-(saddr) saddr=FE20H-FF1FH

Subtract the contents of the short direct memory addressed in the second operand from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUB A, sfr

Function: A, CY ← A-sfr

Subtract the contents of the special function register specified in the second operand from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUB saddr, saddr

Function: (saddr), CY ← (saddr)-(saddr) saddr=FE20H-FE1FH
Subtract the contents of the short direct memory addressed in the second operand from the contents of the short direct memory addressed in the first operand. Set the carry flag a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.
Describe the short direct memory address or label in each of the first and second operands saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUB A, mem

Function: A, CY ← A-(mem)
Subtract the contents of the memory addressed by the memory addressing described in the second operand from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.
If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the subtraction.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUB mem, A

Function: (mem), CY ← (mem)-A

Subtract the A register contents from the contents of the memory addressed by the memory addressing described in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the subtraction.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBC A, #byte

Function: A, CY ← A-byte-CY byte=00H-FFH

Subtract the 8-bit immediate data specified in the second operand and the carry flag from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBC saddr, #byte

Function: (saddr), CY ← (saddr)-byte-CY saddr=FE20H-FF1FH
byte=00H-FFH

Subtract the 8-bit immediate data specified in the second operand and the carry flag from the contents of the short direct memory addressed in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.
Describe the short direct memory address or label in the first operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Caution: If any of the following special function registers is specified in the first operand as short direct memory, the operation result becomes undefined. Do not describe it in the first operand.

Special function register: P4, P5

SUBC sfr, #byte

Function: sfr, CY ← sfr-byte-CY byte=00H-FFH

Subtract the 8-bit immediate data specified in the second operand and the carry flag from the contents of the special function register specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Caution: If any of the following special function registers is specified in the first operand, the operation result becomes undefined. Do not describe it in the first operand.

Special function register: P4, P5, PM5, MM, external SFR

SUBC r, r1

Function: r, CY ← r-r1-CY

Subtract the contents of the 8-bit register specified in the second operand and the carry flag from the contents of the 8-bit register specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBC A, saddr

Function: A, CY ← A-(saddr)-CY saddr=FE20H-FF1FH

Subtract the contents of the short direct memory addressed in the second operand and the carry flag from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBC A, sfr

Function: A, CY ← A-sfr-CY

Subtract the contents of the special function register specified in the second operand and the carry flag from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBC saddr, saddr

Function: (saddr), CY ← (saddr)-(saddr)-CY

saddr=FE20H-FF1FH

Subtract the contents of the short direct memory addressed in the second operand and the carry flag from the contents of the short direct memory addressed in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.
Describe the short direct memory address or label in each of the first and second operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Caution: If any of the following special function registers is specified in the first operand as short direct memory, the operation result becomes undefined. Do not describe it in the first operand.

Special function register: P4, P5

SUBC A, mem

Function: A, CY ← A-(mem)-CY

Subtract the contents of the memory addressed by the memory addressing described in the second operand and the carry flag from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the subtraction.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBC mem, A

Function: (mem), CY ← (mem)-A-CY

Subtract the A register contents and the carry flag from the memory addressed by the memory addressing described in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the subtraction.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

AND A, #byte

Function: A ← A ∧ byte byte=00H-FFH

AND the A register contents with the 8-bit immediate data specified in the second operand and set the result in the A register.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

AND saddr, #byte

Function: (saddr) ← (saddr) ∧ byte saddr=FE20H-FF1FH
byte=00H-FFH

AND the contents of the short direct memory addressed in the first operand with the 8-bit immediate data specified in the second operand and set the result in the short direct memory addressed in the first operand.

Describe the short direct memory address or label in the first operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

AND sfr, #byte

Function: sfr ← sfr ∧ byte byte=00H=FFH

AND the contents of the special function register specified in the first operand with the 8-bit immediate data specified in the second operand and set the result in the special function register specified in the first operand.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

Description example: AND MM, #0FH; Only the high-order four bits of MM register are reset (The low-order four bits do not change)

AND r, r1

Function: r ← r ∧ r1

AND the contents of the 8-bit register specified in the first operand with the contents of the 8-bit register specified in the second operand and set the result in the 8-bit register specified in the first operand.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

AND A, saddr

Function: A ← A ∧ (saddr) saddr=FE20H-FF1FH

AND the A register contents with the contents of the short direct memory addressed in the second operand and set the result in the A register.

Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

AND A, sfr

Function: A ← A ∧ sfr

AND the A register contents with the contents of the special function register specified in the second operand and set the result in the A register.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

AND saddr, saddr

Function: (saddr) ← (saddr) ∧ (saddr) saddr=FE20H-FF1FH

AND the contents of the short direct memory specified in the first operand with the contents of the short direct memory specified in the second operand and set the result in the short direct memory addressed in the first operand.

Describe the short direct memory address or label in each of the first and second operands saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

AND A, mem

Function: A ← A ∧ (mem)

AND the A register contents with the contents of the memory addressed by the memory addressing described in the second operand and set the result in the A register.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the operation.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

AND mem, A

Function: (mem) ← (mem) ∧ A

AND the contents of the memory addressed by the memory addressing described in the first operand with the A register contents and set the results in the memory addressed in the first operand.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the operation.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

OR A, #byte

Function: A ← A V byte byte=00H-FFH

OR the A register contents with the 8-bit immediate data specified in the second operand and set the result in the A register.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

OR saddr, #byte

Function: (saddr) ← (saddr) V byte saddr=FE20H-FF1FH
byte=00H-FFH

OR the contents of the short direct memory addressed in the first operand with the 8-bit immediate data specified in the second operand and set the result in the short direct memory addressed in the first operand.

Caution: Describe the short direct memory address or label in the first operand saddr.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

OR sfr, #byte

Function: sfr ← sfr V byte byte=00H-FFH

OR the contents of the special function register specified in the first operand with the 8-bit immediate data specified in the second operand and set the result in the special function register specified in the first operand.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

Description example: MOV PM1, 00H; Output 1 from the high-order four bits of port 1. (The low-order four bits do not change).

OR P1, #F0H

OR r, r1

Function: r ← r V r1

OR the contents of the 8-bit register specified in the first operand with the contents of the 8-bit register specified in the second operand and set the result in the 8-bit register specified in the first operand.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

OR A, saddr

Function: A ← A V (saddr) saddr=FE20H-FF1FH
OR the A register contents with the contents of the short direct memory addressed in the second operand and set the result in the A register.
Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

OR A, sfr

Function: A ← A V sfr
OR the A register contents with the contents of the special function register specified in the second operand and set the result in the A register.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

OR saddr, saddr

Function: (saddr) ← (saddr) V (saddr) saddr=FE20H-FF1FH
OR the contents of the short direct memory addressed in the first operand with the contents of the short direct memory addressed in the second operand and set the result in the short direct memory addressed in the first operand.
Describe the short direct memory address or label in each of the first and second operands saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

OR A, mem

OR A, mem

Function: A ← A V (mem)

OR the A register contents with the contents of the memory addressed by the memory addressing described in the second operand and set the result in the A register.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the operation.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

OR mem, A

Function: (mem) ← (mem) V A

OR the contents of the memory addressed by the memory addressing described in the first operand with the A register contents and set the result in the memory addressed in the first operand.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the operation.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

XOR A, #byte

Function: A ← A V byte byte=00H-FFH

Exclusive-OR the A register contents with the 8-bit immediate data specified in the second operand and set the result in the A register.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

Description example: XOR A, #0FFH; The contents of A register are inverted.

XOR saddr, #byte

Function: (saddr) ← (saddr) ∀ byte saddr=FE20H-FF1FH
byte=00H-FFH

Exclusive-OR the contents of the short direct memory addressed in the first operand with the 8-bit immediate data specified in the second operand and set the result in the short direct memory addressed in the first operand.

Describe the short direct memory address or label in the first operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

XOR sfr, #byte

Function: sfr ← sfr ∀ byte byte=00H-FFH

Exclusive-OR the contents of the special function register specified in the first operand with the 8-bit immediate data specified in the second operand and set the result in the special function register specified in the first operand.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

XOR r, r1

Function: r ← r ∀ r1

Exclusive-OR the contents of the 8-bit register specified in the first operand with the contents of the 8-bit register specified in the second operand and set the result in the 8-bit register specified in the first operand.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

XOR A, saddr

Function: A ← A ∀ (saddr) saddr=FE20H-FF1FH

Exclusive-OR the A register contents with the contents of the short direct memory addressed in the second operand and set the result in the A register. Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

XOR A, sfr

Function: A ← A ∀ sfr

Exclusive-OR the A register contents with the contents of the special function register specified in the second operand and set the result in the A register.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

XOR saddr, saddr

Function: (saddr) ← (saddr) ∀ (saddr) saddr=FE20H-FF1FH
Exclusive-OR the contents of the short direct memory addressed in the first operand with the contents of the short direct memory addressed in the second operand and set the result in the short direct memory addressed in the first operand.
Describe the short direct memory address or label in each of the first and second operands saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

XOR A, mem

Function: A ← A ∀ (mem)
Exclusive-OR the A register contents with the contents of the memory addressed by the memory addressing described in the second operand and set the result in the A register.
If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the operation.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

XOR mem, A

Function: (mem) ← (mem) ∀ A

Exclusive-OR the contents of the memory addressed by the memory addressing described in the first operand with the A register contents and set the result in the memory addressed in the first operand.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the operation.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

CMP A, #byte

Function: A-byte byte=00H-FFH

Subtract the 8-bit immediate data specified in the second operand from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMP saddr, #byte

Function: (saddr)-byte

saddr=FE20H-FF1FH
byte=00H-FFH

Subtract the 8-bit immediate data specified in the second operand from the contents of the short direct memory addressed in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.
After the CMP instruction is executed, the short direct memory contents do not change.
Describe the short direct memory address or label in the first operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMP sfr, #byte

Function: sfr-byte

byte=00H-FFH

Subtract the 8-bit immediate data specified in the second operand from the contents of the special function register specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.
After the CMP instruction is executed, the special function register contents do not change.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMP r, r1

Function: r-r1

Subtract the contents of the 8-bit register specified in the second operand from the contents of the 8-bit register specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMP instruction is executed, the 8-bit register r and r1 contents do not change.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMP A, saddr

Function: A-(saddr) saddr=FE20H-FF1FH

Subtract the contents of the short direct memory addressed in the second operand from the A register contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMP instruction is executed, the A register and short direct memory contents do not change.

Describe the short direct memory address or label in the second operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMP A, sfr

Function: A-sfr

Subtract the contents of the special function register specified in the second operand from the A register contents. Set the carry flag if a carry is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMP instruction is executed, the A register and special function register contents do not change.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMP saddr, saddr

Function: (saddr)-(saddr) saddr=FE20H-FF1FH

Subtract the contents of the short direct memory addressed in the second operand from the contents of the short direct memory addressed in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction.

After the CMP instruction is executed, the short direct memory contents do not change.

Describe the short direct memory address or label in each of the first and second operands saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMP A, mem

Function: A-(mem)

Subtract the contents of the memory addressed by the memory addressing described in the second operand from the A register. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the operation.

After the CMP instruction is executed, the A register and memory contents do not change.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMP mem, A

Function: (mem)-A

Subtract the A register contents from the memory addressed by the memory addressing described in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

If auto increment ([DE+] or [HL+]) or auto decrement ([DE-] or [HL-]) is specified as mem, automatically increment or decrement the register pair DE or HL contents by one after the operation.

After the CMP instruction is executed, the A register and memory contents do not change.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

18.6.4 16-bit arithmetic and logical instructions

ADDW AX, #word

Function: AX, CY ← AX+word word=0000H-FFFFH

Add the 16-bit immediate data specified in the second operand to the register pair AX contents in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDW saddrp, #word

Function: (saddrp), CY ← (saddrp)+word saddrp=FE20H-FF1EH
word=0000H-FFFFH

Add the 16-bit immediate data specified in the second operand to the contents of the 2-byte area of the short direct memory addressed in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Describe the short direct memory address or label in the first operand saddrp as it is (even address only).

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDW sfrp, #word

Function: sfrp, CY ← sfrp+word word=0000H-FFFFH
Add the 16-bit immediate data specified in the second operand to the contents of the special function register specified in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDW rp, rp1

Function: rp, CY ← rp+rp1
Add the contents of the 16-bit register pair specified in the second operand to the contents of the 16-bit register pair specified in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDW AX, saddrp

Function: AX, CY ← AX+(saddrp) saddrp=FE20H-FF1EH
Add the contents of the 2-byte area of the short direct memory addressed in the second operand to the register pair AX contents in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Describe the short direct memory address or label in the second operand saddrp as it is (even address only).

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDW AX, sfrp

Function: AX, CY ← AX+sfrP

Add the contents of the 16-bit special function register specified in the second operand to the registr pair AX contents in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

ADDW saddrp, saddrp

Function: (saddrp), CY ← (saddrp)+(saddrp)

saddrp=FE20H-FF1EH

Add the contents of the 2-byte area of the short direct memory addressed in the second operand to the contentes of the 2-byte area of the short direct memory addressed in the first operand in binary. Set the carry flag if a carry is generated as a result of the addition. Reset the carry flag if no carry is generated.

Describe the short direct memory address or label in each of the first and second operands saddrp as it is (even address only).

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBW AX, #word

Function: AX, CY ← AX-word word=0000H-FFFFH

Subtract the 16-bit immediate data specified in the second operand from the register pair AX contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBW saddrp, #word

Function: (saddrp), CY ← (saddrp)-word saddrp=FE20H-FF1EH
word=0000H-FFFFH

Subtract the 16-bit immediate data specified in the second operand from the contents of the 2-byte area of the short direct memory addressed in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Describe the short direct memory address or label in the first operand saddrp as it is (even address only).

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBW sfrp, #word

Function: sfrp, CY — sfrp-word word=0000H-FFFFH
Subtract the 16-bit immediate data specified in the second operand from the contents of the 16-bit special function register specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBW rp, rp1

Function: rp, CY ← rp-rp1
Subtract the contents of the 16-bit register pair specified in the second operand from the contents of the 16-bit register pair specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBW AX, saddrp

Function: AX, CY ← AX-(saddrp) saddrp=FE20H-FF1EH
Subtract the contents of the 2-byte area of the short direct memory addressed in the second operand from the register pair AX contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.
Describe the short direct memory address or label in the second operand saddrp as it is (even address only).

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBW AX, sfrp

Function: AX, CY ← AX-(sfrp)

Subtract the contents of the 16-bit special function register specified in the second operand from the register pair AX contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

SUBW saddrp, saddrp

Function: (saddrp), CY ← (saddrp)-(saddrp)

saddrp=FE20H-FF1EH

Subtract the contents of the 2-byte area of the short direct memory specified in the second operand from the contents of the 2-byte area of the short direct memory specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

Describe the short direct memory address or label in each of the first and second operands saddrp as it is (even address only).

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPW AX, #word

Function: AX-word ← word=0000H-FFFFH

Subtract the 16-bit immediate data specified in the second operand from the register pair AX contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMPW instruction is executed, the register pair AX contents do not change.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPW saddrp, #word

Function: (saddrp)-word saddrp=FE20H-FF1EH
word=0000H-FFFFH

Subtract the 16-bit immediate data specified in the second operand from the contents of the short direct memory addressed in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMPW instruction is generated, the short direct memory contents do not change.

Describe the short direct memory address or label in the first operand saddrp as it is (even address only).

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Description example:

```
CMPW 0FE50H, #8000H ; Branch to address
BGT $JMP              indicated by label
                      JMP if the contents
                      of memory addresses
                      FE51H and FE50H are
                      greater than 8000H.
```

CMPW sfrp, #word

Function: sfrp-word word=0000H-FFFFH

Subtract the 16-bit immediate data specified in the second operand from the contents of the 16-bit special function register specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMPW instruction is executed, the contents of the 16-bit special function register specified in the first operand do not change.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPW rp, rp1

Function: rp-rp1

Subtract the contents of the 16-bit registr pair specified in the second operand from the contents of the 16-bit register pair specified in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMPW instruction is executed, the contents of the register pairs specified in the first and second operands do not change.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPW AX, saddrp

Function: AX-(saddrp) saddrp=FE20H-FF1EH

Subtract the contents of the 2-byte area of the short direct memory specified in the second operand from the register pair AX contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMPW instruction is executed, the register pair and short direct memory contents do not change.

Describe the short direct memory address or label in the second operand saddrp as it is (even address only).

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPW AX, sfrp

Function: AX-sfrp

Subtract the contents of the 16-bit special function register specified in the second operand from the register pair AX contents. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMPW instruction is executed, the contents of the register pair AX and the 16-bit special function register specified in the second operand do not change.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPW saddrp, saddrp

Function: (saddrp)-(saddrp) saddrp=FE20H-FF1EH

Subtract the contents of the 2-byte area of the short direct memory addressed in the second operand from the contents of the 2-byte area of the short direct memory addressed in the first operand. Set the carry flag if a borrow is generated as a result of the subtraction. Reset the carry flag if no borrow is generated.

After the CMPW instruction is executed, the short direct memory contents do not change.

Describe the short direct memory address or label in each of the first and second operands saddrp as it is (even address only).

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

18.6.5 Multiplication and division instructions

MULU r1

Function: AX ← A x r1
Multiply the A register contents by the contents of the 8-bit register specified in the operand and set the result in the register pair AX.

Flag operation: No change

DIVUW r1

Function: AX, r1 ← AX ÷ 1
Divide the register pair AX contents by the contents of the 8-bit register specified in the operand and set the quotient in the register pair AX and the remainder in the register specified in the operand.

Flag operation: No change

MULUW rp1

Function: AX, rp1 ← AX x rp1
Multiply the registr pair AX contents by the contents of the 16-bit register pair specified in the operand and set the high-order 16 bits of the result in the register pair AX and the low-order 16 bits in the 16-bit register pair specified in the operand.

Flag operation: No change

DIVUX rp1

Function: AXDE, rp1 ← AXDE ÷ rp1

Divide the contents of the register pair AX (high-order 16 bits) and register pair DE (low-order 16 bits), (32-bit data), by the contents of the 16-bit register pair specified in the operand and set the quotient in the register pair AX (high-order 16 bits) and register pair DE (low-order 16 bits) and the remainder in the 16-bit register pair specified in the operand.

Flag operation: No change

18.6.6 Signed multiplication instruction

MULW rp1

Function: AX, rp1 ← AX x rp1 (signed)
Multiply the register pair AX contents by the contents of the 16-bit register pair specified in the operand with sign and set the high-order 16 bits of the result in the register pair AX and the low-order 16 bits in the 16-bit register pair specified in the operand.

Flag operation: No change

18.6.7 Increment and decrement instructions

INC r1

Function: r1 ← r1+1
Increment the contents of the 8-bit register specified in the operand by one.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	

INC saddr

Function: (saddr) ← (saddr) +1 saddr=FE20H-FF1FH
Increment the contents of the short direct memory addressed in the operand by one.
Describe the short direct memory address or label in the operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	

Description example: INC, TB1;Add one to the contents of the short direct memory addressed by label TB1.

DEC r1

Function: r1 ← r1-1
Decrement the contents of the 8-bit register specified in the operand by one.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	

DEC saddr

Function: (saddr) ← (saddr) -1 saddr=FE20H-FF1FH
Decrement the contents of the short direct memory addressed in the operand by one.
Describe the short direct memory address or label in the operand saddr as it is.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	

INCW rp2

Function: rp2 ← rp2+1
Increment the contents of the 16-bit register pair specified in the operand by one.

Flag operation: No change

Description example: INCW DE;DE ← DE+1

INCW saddrp

Function: (saddrp) ← (saddrp)+1 saddrp=FE20H-FF1EH
Increment the contents of the 2-byte area of the short direct memory addressed in the operand by one
Describe the short direct memory address or label in the operand saddr as it is (even address only).

Flag operation: No change

DECW rp2

Function: rp2 ← rp2-1
Decrement the contents of the 16-bit register pair specified in the operand by one.

Flag operation: No change

Description example: DECW HL;HL ← HL-1

DECW saddrp

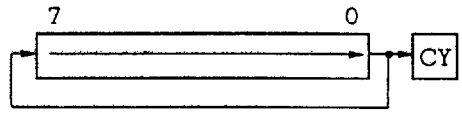
Function: (saddrp) ← (saddrp)-1 saddrp=FE20H-FF1EH
Decrement the contents of the 2-byte area of the short direct memory addressed in the operand by one. Describe the short direct memory address or label in the operand saddrp as it is (even address only).

Flag operation: No change

18.6.8 Shift and rotate instructions

ROR rl, n

Function: $(CY, rl_7 \leftarrow rl_0, rl_{m-1} \leftarrow rl_m) \times n$ n=0-7



Rotate the contents of the 8-bit register specified in the first operand right as many bits as specified by the 3-bit immediate data described in the second operand. Transfer the LSB contents of the 8-bit register to the MSB and also set the LSB contents in the carry flag. (However, when n=0, the operation is not performed.)

Flag operation:

S	Z	AC	P/V	CY
			P	x

ROL rl, n

Function: $(CY, rl_0 \leftarrow rl_7, rl_{m+1} \leftarrow rl_m) \times n$ n=0-7

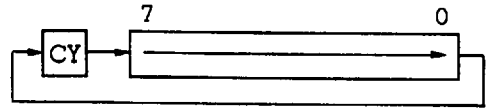
Rotate the contents of the 8-bit register specified in the first operand left as many bits as specified by the 3-bit immediate data described in the second operand. Transfer the MSB contents of the 8-bit register to the LSB and also set the MSB contents in the carry flag. (However, when n=0, the operation is not performed.)

Flag operation:

S	Z	AC	P/V	CY
			P	x

RORC r1, n

Function: (CY ← $r1_0$, $r1_7$ ← CY, $r1_{m-1}$ ← $r1_m$) x n n=0-7



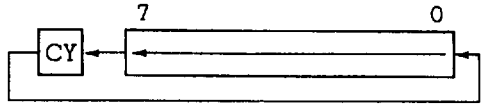
Rotate the contents of the 8-bit register specified in the first operand right as many bits as specified by the 3-bit immediate data described in the second operand through the carry flag.

Flag operation:

S	Z	AC	P/V	CY
			P	x

ROLC r1, n

Function: (CY ← $r1_7$, $r1_0$ ← CY, $r1_{m+1}$ ← $r1_m$) x n n=0-7



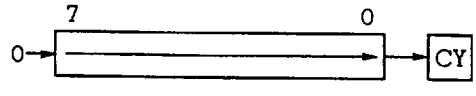
Rotate the contents of the 8-bit register specified in the first operand left as many bits as specified by the 3-bit immediate data described in the second operand through the carry flag.

Flag operation:

S	Z	AC	P/V	CY
			P	x

SHR r1, n

Function: (CY ← $r1_0$, $r1_7$ ← 0, $r1_{m-1}$ ← $r1_m$) x n n=0-7



Shift the contents of the 8-bit register specified in the first operand right as many bits as specified by the 3-bit immediate data described in the second operand. Shift the LSB contents of the 8-bit register to the carry flag and set 0 in the MSB. (However, when n=0, the operation is not performed.)

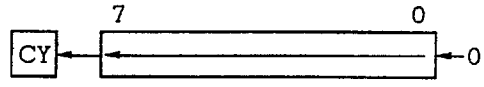
Flag operation:

S	Z	AC	P/V	CY
x	x	0	P	x

Description example: SHR R1, 1;Halve the A register contents (set the remainder in CY).

SHL r1, n

Function: (CY $\leftarrow$ $r1_7$, $r1_0 \leftarrow 0$, $r1_{m+1} \leftarrow r1_m$) x n n=0-7



Shift the contents of the 8-bit register specified in the first operand left as many bits as specified by the 3-bit immediate data described in the second operand. Shift the MSB contents of the 8-bit register to the carry flag and set 0 in the LSB. (However, when n=0, the operation is not performed)

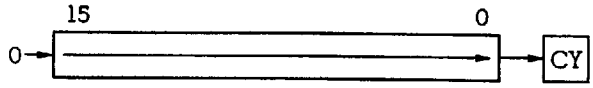
Flag operation:

S	Z	AC	P/V	CY
x	x	0	P	x

Description example: SHL R14, 3;Left the L register contents left three bits. Set the bit 5 contents before the shift in the carry flag.

SHRW rp1, n

Function: (CY $\leftarrow$ $rp1_0$, $rp1_{15} \leftarrow 0$, $rp1_{m-1} \leftarrow rp1_m$) x n n=0-7



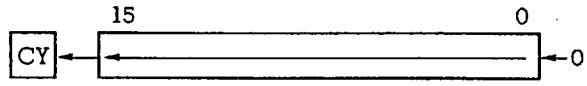
Shift the contents of the 16-bit register pair specified in the first operand right as many bits as specified by the 3-bit immediate data described in the second operand. Shift the LSB contents of the 16-bit register pair to the carry flag and set 0 in the MSB. (However, when n=0, the operation is not performed.)

Flag operation:

S	Z	AC	P/V	CY
x	x	0	P	x

SHLW rpl, n

Function: (CY ← rpl_{15}, rpl_0 ← 0, rpl_{m+1} ← rpl_m) x n n=0-7



Shift the contents of the 16-bit register pair specified in the first operand left as many bits as specified by the 3-bit immediate data described in the second operand. Shift the MSB contents of the 16-bit register pair to the carry flag and set 0 in the LSB. (However, when n=0, the operation is not performed.)

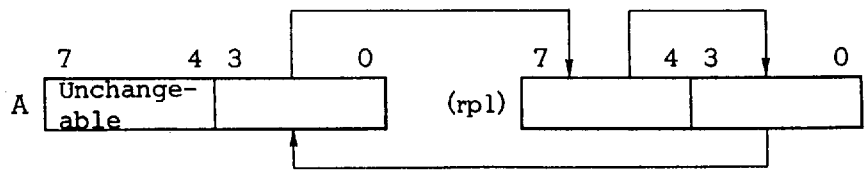
Flag operation:

S	Z	AC	P/V	CY
x	x	0	P	x

Description example: SHLW RP0, 1;Double the AX register pair contents.

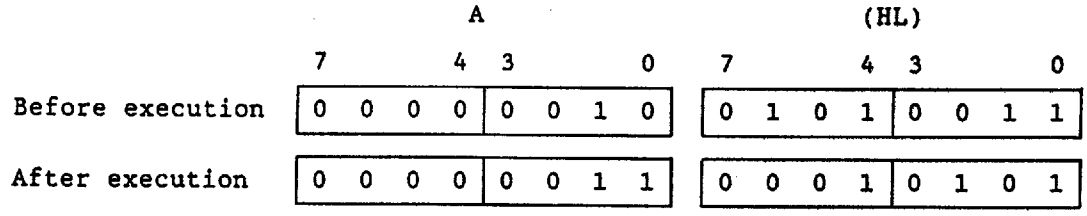
ROR4 [rpl]

Function: A_{3-0} ← $(rpl)_{3-0}$, $(rpl)_{7-4}$ ← A_{3-0}, $(rpl)_{3-0}$ ← $(rpl)_{7-4}$



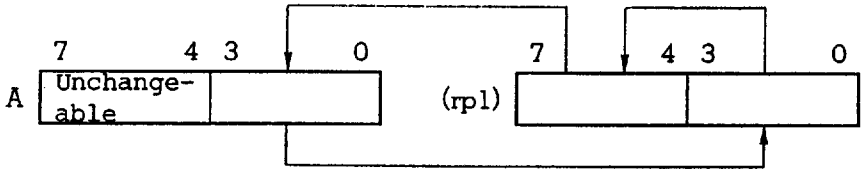
Rotate the low-order four bits of the A register and the high-order four bits and low-order four bits of the memory addressed in the operand right four bits. (A register bits 7-4 are not affected.)

Description example: ROR4[HL];



ROL4 [rp1]

Function: $A_{3-0} \leftarrow (rp1)_{7-4}$, $(rp1)_{3-0} \leftarrow A_{3-0}$, $(rp1)_{7-4} \leftarrow (rp1)_{3-0}$



Rotate the low-order four bits of the A register and the high-order four bits and low-order four bits of the memory addressed in the operand left four bits. (A register bits 7-4 are not affected.)

Flag operation: No change

18.6.9 BCD adjustment instructions

ADJBA

Function: Judge the contents of the A register, carry flag (CY), and auxiliary carry flag (AC) and make decimal adjustments as listed in Table 18-7.
This instruction is significant only after decimal (BCD) data pieces are added together.

Table 18-7 Decimal Data Adjustment (ADJBA Instruction)

Condition		Operation	After adjustment	
			CY	AC
$A_{3-0} \leq 9$ AC=0	$A_{7-4} \leq 9$ and CY=0	A ← A	0	0
	$A_{7-4} \geq 10$ or CY=1	A ← A+60H	1	0
$A_{3-0} \geq 10$ AC=0	$A_{7-4} < 9$ and CY=0	A ← A+06H	0	1
	$A_{7-4} \leq 9$ or CY=1	A ← A+66H	1	1
AC=1	$A_{7-4} \geq 9$ and CY=0	A ← A+06H	0	1
	$A_{7-4} \geq 10$ or CY=1	A ← A+66H	1	1

Flag operation:

S	Z	AC	P/V	CY
x	x	x	P	x

ADJBS

Function: Judge the contents of the A register, carry flag (CY), and auxiliary carry flag (AC) and make decimal adjustments as listed in Table 18-8.
This instruction is significant only after decimal (BCD) data pieces are subtracted together.

Table 18-8 Decimal Data Adjustment (ADJBS Instruction)

Condition		Operation	After adjustment	
			CY	AC
$A_{3-0} \leqq 9$ AC=0	$A_{7-4} \leqq 9$ and CY=0	A ← A	0	0
	$A_{7-4} \geqq 10$ or CY=1	A ← A-60H	1	0
$A_{3-0} \geqq 10$ AC=0	$A_{7-4} < 9$ and CY=0	A ← A-06H	0	1
	$A_{7-4} \geqq 9$ or CY=1	A ← A-66H	1	1
AC=1	$A_{7-4} \leqq 9$ and CY=0	A ← A-06H	0	1
	$A_{7-4} \geqq 10$ or CY=1	A ← A-66H	1	1

Flag operation:

S	Z	AC	P/V	CY
x	x	x	P	x

18.6.10 Data conversion instruction

CVTBW

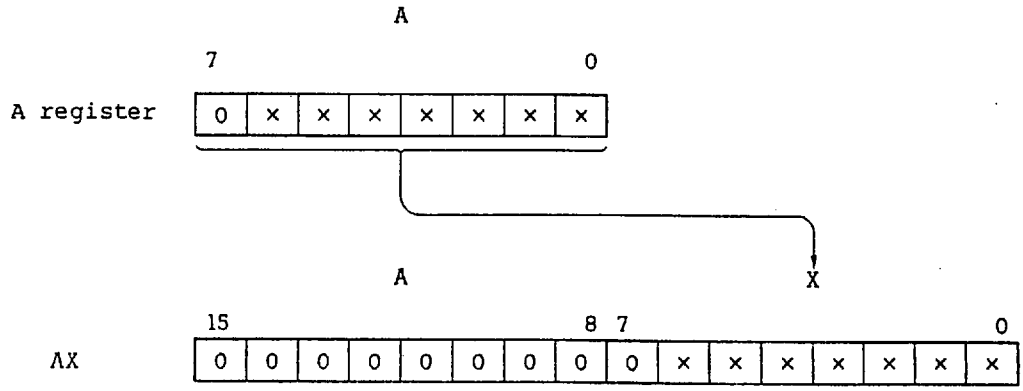
Function:
- When A_7=0
  X ← A, A ← 00H
- When A_7=1
  X ← A, A ← FFH

Extend the signed 8-bit data in the A register to the signed 16-bit data in the AX register. (See Fig. 18-10)

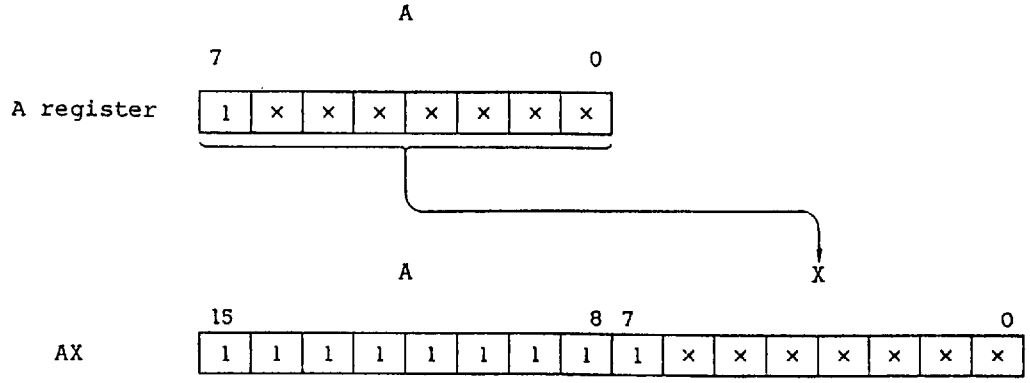
Flag operation: No change

Fig. 18-10 Data Conversion by CVTBW Instruction

(a) When A_7=0



(b) When A_7=1



18.6.11 Bit manipulation instructions

MOV1 CY, saddr.bit

Function: CY ← (saddr.bit) saddr=FE20H-FF1FH
bit=0-7

Transfer the contents of the short direct memory bit addressed in the second operand to the carry flag. Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation:

S	Z	AC	P/V	CY
				x

MOV1 CY, sfr.bit

Function: CY ← sfr.bit bit=0-7

Transfer the contents of the bit of the special function register specified in the second operand, addressed by the 3-bit immediate data to the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

MOV1 CY, A.bit

Function: CY ← A.bit bit=0-7

Transfer the contents of the A register bit addressed by the 3-bit immediate data in the second operand to the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

MOV1 CY, X.bit

Function: CY ← X.bit bit=0-7

Transfer the contents of the X register bit addressed by the 3-bit immediate data in the second operand to the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

MOV1 CY, PSWH.bit

Function: CY ← PSW_H.bit bit=0-7

Transfer the contents of the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand to the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

MOV1 CY, PSWL.bit

Function: CY ← PSW_L.bit bit=0-7

Transfer the contents of the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand to the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

MOV1 saddr.bit, CY

Function: (saddr.bit) ← CY saddr=FE20H-FF1FH
bit=0-7

Transfer the carry flag contents to the short direct memory bit addressed in the first operand.
Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation: No change

MOV1 sfr.bit, CY

Function: sfr.bit ← CY bit=0-7

Transfer the carry flag contents to the bit of the special function register specified in the first operand, addressed by the 3-bit immediate data.

Flag operation: No change

MOV1 A.bit, CY

Function: A.bit ← CY bit=0-7

Transfer the carry flag contents to the A register bit addressed by the 3-bit immediate data in the first operand.

Flag operation: No change

MOV1 X.bit, CY

Function: X.bit ← CY bit=0-7

Transfer the carry flag contents to the X register bit addressed by the 3-bit immediate data in the first operand.

Flag operation: No change

MOV1 PSWH.bit, CY

Function: PSW_H.bit ← CY bit=0-7

Transfer the carry flag contents to the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand.

Flag operation: No change

MOV1 PSWL.bit, CY

Function: PSW_L.bit ← CY bit=0-7

Transfer the carry flag contents to the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand.

Flag operation: No change

AND1 CY, saddr.bit

Function: CY ← CY ∧ (saddr.bit) saddr=FE20H-FF1FH
bit=0-7

AND the carry flag contents with the contents of the short direct memory bit addressed in the second operand and set the result in the carry flag.
Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, /saddr.bit

Function: CY ← CY ∧ ($\overline{\text{saddr.bit}}$) saddr=FE20H-FF1FH
bit=0-7

AND the carry flag contents with the inversion contents of the short direct memory bit addressed in the second operand and set the result in the carry flag. Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, sfr.bit

Function: CY ← CY ∧ sfr.bit bit=0-7

AND the carry flag contents with the contents of the bit of the special function register specified in the second operand, addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, /sfr.bit

Function: CY ← CY ∧ $\overline{\text{sfr.bit}}$ bit=0-7

AND the carry flag contents with the inversion contents of the bit of the special function register specified in the second operand, addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, A.bit

Function: CY ← CY ∧ A.bit bit=0-7

AND the carry flag contents with the contents of the A register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, /A.bit

Function: CY ← CY ∧ $\overline{\text{A.bit}}$ bit=0-7

AND the carry flag contents with the inversion contents of the A register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, X.bit

Function: CY ← CY ∧ X.bit bit=0-7

AND the carry flag contents with the contents of the X register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, /X.bit

Function: CY ← CY ∧ $\overline{X.bit}$ bit=0-7

AND the carry flag contents with the inversion contents of the X register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, PSWH.bit

Function: CY ← CY ∧ PSW_H.bit bit=0-7

AND the carry flag contents with the contents of the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, /PSWH.bit

Function: CY ← CY ∧ $\overline{PSW_H.bit}$ bit=0-7

AND the carry flag contents with the inversion contents of the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, PSWL.bit

Function: CY ← CY ∧ PSW_L.bit bit=0-7

AND the carry flag contents with the contents of the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

AND1 CY, /PSWL.bit

Function: CY ← CY ∧ $\overline{PSW_L.bit}$ bit=0-7

AND the carry flag contents with the inversion contents of the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, saddr.bit

Function: CY ← CY V (saddr.bit) saddr=FE20H-FF1FH
bit=0-7

OR the carry flag contents with the contents of the short direct memory bit addressed in the second operand and set the result in the carry flag.
Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, /saddr.bit

Function: CY ← CY V ($\overline{\text{saddr.bit}}$) saddr=FE20H-FF1FH
bit=0-7

OR the carry flag contents with the inversion contents of the short direct memory bit addressed by the 3-bit immediate data in the second operand and the result in the carry flag.

Caution: Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, sfr.bit

Function: CY ← CY V sfr.bit bit=0-7

OR the carry flag contents with the contents of the bit of the special function register specified in the second operand, addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, /sfr.bit

Function: CY ← CY V $\overline{\text{sfr.bit}}$ bit=0-7

OR the carry flag contents with the inversion contents of the bit of the special function register specified in the second operand, addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, A.bit

Function: CY ← CY ∧ A.bit bit=0-7

OR the carry flag contents with the contents of the A register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, /A.bit

Function: CY ← CY V $\overline{\text{A.bit}}$ bit=0-7

OR the carry flag contents with the inversion contents of the A register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, X.bit

Function: CY ← CY V X.bit bit=0-7

OR the carry flag contents with the contents of the X register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, /X.bit

Function: CY ← CY V $\overline{X.bit}$ bit=0-7

OR the carry flag contents with the inversion contents of the X register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, PSWH.bit

Function: CY ← CY V PSW_H.bit bit=0-7

OR the carry flag contents with the contents of the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, /PSWH.bit

Function: CY ← CY $\overline{PSW_H.bit}$ bit=0-7
OR the carry flag contents with the inversion contents of the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, PSWL.bit

Function: CY ← CY $PSW_L.bit$ bit=0-7
OR the carry flag contents with the contents of the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

OR1 CY, /PSWL.bit

Function: CY ← CY V $\overline{PSW_L.bit}$ bit=0-7
OR the carry flag contents with the inversion contents of the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

XOR1 CY, saddr.bit

Function: CY ← CY ∀ (saddr.bit) saddr=FE20H-FF1FH
bit=0-7

Exclusive-OR the carry flag contents with the contents of the short direct memory bit addressed in the second operand and set the result in the carry flag. Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation:

S	Z	AC	P/V	CY
				x

XOR1 CY, sfr.bit

Function: CY ← CY ∀ sfr.bit bit=0-7

Exclusive-OR the carry flag contents with the contents of the bit of the special function register specified in the second operand, addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

XOR1 CY, A.bit

Function: CY ← CY ∀ A.bit bit=0-7

Exclusive-OR the carry flag contents with the contents of the A register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

XOR1 CY, X.bit

Function: CY ← CY ∀ X.bit bit=0-7

Exclusive-OR the carry flag contents with the contents of the X register bit addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

XOR1 CY, PSWH.bit

Function: CY ← CY ∀ PSW_H.bit bit=0-7

Exclusive-OR the carry flag contents with the contents of the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

XOR1 CY, PSWL.bit

Function: CY ← CY ∀ PSW_L.bit bit=0-7

Exclusive-OR the carry flag contents with the contents of the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the second operand and set the result in the carry flag.

Flag operation:

S	Z	AC	P/V	CY
				x

SET1 saddr.bit

Function: (saddr.bit) ← 1 saddr=FE20H-FF1FH
bit=0-7

Set the short direct memory bit addressed in the operand to 1.
Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation: No change

SET1 sfr.bit

Function: sfr.bit ← 1 bit=0-7

Set the bit of the special function register specified in the operand, addressed by the 3-bit immediate data in the operand to 1.

Flag operation: No change

SET1 A.bit

Function: A.bit ← 1 bit=0-7

Set the A register bit addressed by the 3-bit immediate data in the operand to 1.

Flag operation: No change

SET1 X.bit

Function: X.bit ← 1 bit=0-7

Set the X register bit addressed by the 3-bit immediate data in the operand to 1.

Flag operation: No change

SET1 PSWH.bit

Function: PSW_H.bit ← 1 bit=0-7
Set the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the operand to 1.

Flag operation: No change

SET1 PSWL.bit

Function: PSW_L.bit ← 1 bit=0-7
Set the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the operand to 1.

Flag operation: No change

SET1 CY

Function: CY ← 1
Set the carry flag to 1.

Flag operation:

S	Z	AC	P/V	CY
				1

CLR1 saddr.bit

Function: (saddr.bit) ← 0 saddr=FE20H-FF1FH
bit=0-7
Reset the short direct memory bit addressed in the operand to 0.
Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation: No change

CLR1 sfr.bit

Function: sfr.bit ← 0 bit=0-7
Reset the bit of the special function register specified in the operand, addressed by the 3-bit immediate data in the operand to 0.

Flag operation: No change

CLR1 A.bit

Function: A.bit ← 0 bit=0-7
Reset the A register bit addressed by the 3-bit immediate data in the operand to 0.

Flag operation: No change

CLR1 X.bit

Function: X.bit ← 0 bit=0-7
Reset the X register bit addressed by the 3-bit immediate data in the operand to 0.

Flag operation: No change

CLR1 PSWH.bit

Function: PSW_H.bit ← 0 bit=0-7
Reset the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the operand to 0.

Flag operation: No change

CLR1 PSWL.bit

Function: PSW_L.bit ← 0 bit=0-7

Reset the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the operand to 0.

Flag operation: The flag addressed in the operand is reset to 0.

CLR1 CY

Function: CY ← 0

Reset the carry flag to 0.

Flag operation:

S	Z	AC	P/V	CY
				0

NOT1 saddr.bit

Function: (saddr.bit) ← ($\overline{\text{saddr.bit}}$) saddr=FE20H-FF1FH
bit=0-7

Invert the contents of the short direct memory bit addressed in the operand.

Describe the short direct memory bit address or label in the operand saddr.bit as it is.

Flag operation: No change

NOT1 sfr.bit

Function: sfr.bit ← $\overline{\text{sfr.bit}}$ bit=0-7

Invert the contents of the bit of the special function register specified in the operand, addressed by the 3-bit immediate data in the operand.

Flag operation: No change

NOT1 A.bit

Function: $A.bit \leftarrow \overline{A.bit}$ bit=0-7
Invert the contents of the A register bit addressed by the 3-bit immediate data in the operand.

Flag operation: No change

NOT1 X.bit

Function: $X.bit \leftarrow \overline{X.bit}$ bit=0-7
Invert the contents of the X register bit addressed by the 3-bit immediate data in the operand.

Flag operation: No change

NOT1 PSWH.bit

Function: $PSW_H.bit \leftarrow \overline{PSW_H.bit}$ bit=0-7
Invert the contents of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the operand.

Flag operation: No change

NOT1 PSWL.bit

Function: $PSW_L.bit \leftarrow \overline{PSW_L.bit}$ bit=0-7
Invert the contents of the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the operand.

Flag operation: The flag addressed in the operand is inverted.

NOT1 CY

Function: CY ← $\overline{CY}$

Invert the carry flag contents.

Flag operation:

S	Z	AC	P/V	CY
				x

18.6.12 Call and return instructions

CALL !addr16

Function: $(SP-1) \leftarrow (PC+3)_H$, $(SP-2) \leftarrow (PC+3)_L$, $PC \leftarrow addr16$, $SP \leftarrow SP-2$

addr16=0000H-FDFFH

Save the top address of the next instruction (return address) in the stack memory addressed by the stack pointer (SP) and decrement the SP, then branch to the address addressed by the 3-bit immediate data specified in the operand.

Caution: No instruction can be fetched from addresses FE00H-FFFFH. Do not describe any of the addresses in addr16.

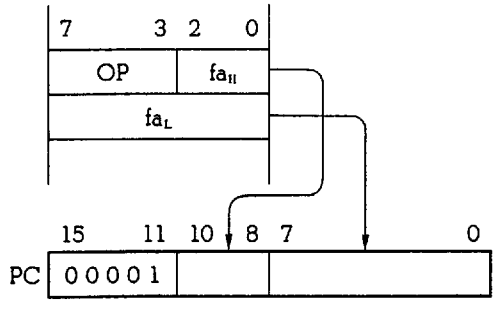
Flag operation: No change

CALLF !addr11

Function: $(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$, $PC_{15-11} \leftarrow 00001$, $PC_{10-0} \leftarrow fa$, $SP \leftarrow SP-2$

addr11=0800H-0FFFH

Fig. 18-11 Data Flow when CALLF Instruction is Executed



Save the top address of the next instruction (return address) in the stack memory addressed by the stack pointer (SP) and decrement the SP, then branch to the address addressed by the effective address made up of 11-bit immediate data fa in the operation code.

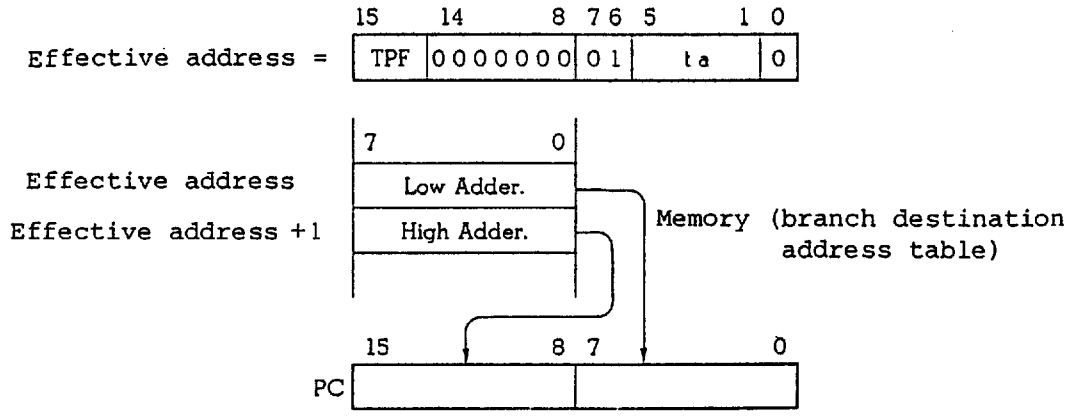
The call enable range is addresses 0800H-0FFFH only. Describe the branch destination address directly with a label or numeric value in the operand addr11 by considering the entry address range.

Flag operation: No change

CALLT [addr5]

Function: $(SP-1) \leftarrow (PC+1)_H$, $(SP-2) \leftarrow (PC+1)_L$, $PC_H \leftarrow$ (TPF, 000000001, ta, 1), $PC_L \leftarrow$ (TPF, 000000001, ta, 0), $SP \leftarrow SP-2$
addr5=40H-7EH

Fig. 18-12 Data Flow when CALLT Instruction is Executed



Save the top address of the next instruction (return address) in the stack memory addressed by the stack pointer (SP) and decrement the SP, then set the contents of the memory (branch destination address table) addressed by the effective address made up of 5-bit immediate data ta in the operation code in the program counter (PC) and branch to the address indicated by the memory contents.

The branch destination address table must be placed in addresses 0040H-007FH.
Describe the branch destination address table address directly with a label or numeric value in the operand addr5.

Remarks: The branch destination address table can be placed in the external memory area (8040H-807FH) by setting the TPF flag to 1.

Flag operation: No change

Description example: CALL [TBL1];Branch to the address addressed by the contents of the table specified by label TBL1.

CALL rp1

Function: $(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$,
$PC_H \leftarrow rp1_H$, $PC_L \leftarrow rp1_L$, $SP \leftarrow SP-2$

Save the top address of the next instruction (return address) in the stack memory addressed by the stack pointer (SP) and decrement the SP, then set the contents of the 16-bit register pair specified in the operand in the program counter (PC) for a branch.

Flag operation: No change

CALL [rp1]

Function: $(SP-1) \leftarrow (PC+2)_H$, $(SP-2) \leftarrow (PC+2)_L$,
$PC_H \leftarrow (rp1+1)$, $PC_L \leftarrow (rp1)$, $SP \leftarrow SP-2$

Save the top address of the next instruction (return address) in the stack memory addressed by the stack pointer (SP) and decrement the SP, then set the contents of the 2-bit area of the memory addressed by the contents of the 16-bit register pair specified in the operand in the program counter (PC) for a branch.

Flag operation: No change

BRK

Function: $(SP-1) \leftarrow PSW_H$, $(SP-2) \leftarrow PSW_L$, $(SP-3) \leftarrow (PC+1)_H$, $(SP-4) \leftarrow (PC+1)_L$, $PC_L \leftarrow (003EH)$, $PC_H \leftarrow (003FH)$, $SP \leftarrow SP-4$, $IE \leftarrow 0$

Save the top address of the next instruction (return address) and the program status word (PSW in the stack memory addressed by the stack pointer and decrement the SP, then set the BRK instruction branch destination address table (003EH and 003FH) contents in the program counter (PC) for branch. Reset the IE flag to 0 to disable the subsequent maskable interrupt requests. (The BRK instruction is acknowledged even in the DI state (IE=0).

Flag operation: No change

RET

Function: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$, $SP \leftarrow SP+2$

Restore the contents of the stack memory addressed by the stack pointer (SP) to the program counter, then increment the SP contents.

Flag operation: No change

RETB

Function: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$,
$PSW_L \leftarrow (SP+2)$, $PSW_H \leftarrow (SP+3)$, $SP \leftarrow SP+4$
Restore the contents of the stack memory addressed by the stack pointer (SP) to the program counter (PS) and program status word (PSW), then increment the SP contents.
The RETB instruction is used to return from BRK instruction or operation code trap.

Flag operation:

S	Z	AC	P/V	CY
R	R	R	R	R

Caution: To return from the interrupt service routine accompanying BRK instruction or operation code trap, be sure to use the RETB instruction.
If the RETI instruction is used, the interrupt control circuit does not operate normally.

RETI

Function: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$, $PSW_L \leftarrow (SP+2)$,
$PSW_H \leftarrow (SP+3)$, $SP \leftarrow SP+4$
Restore the contents of the stack memory addressed by the stack pointer (SP) to the program counter (PC) and program status word (PSW), then increment the SP contents.
The RETI instruction is used to return from interrupt service routine.

Flag operation:

S	Z	AC	P/V	CY
R	R	R	R	R

Caution: To return from the interrupt service routine accompanying BRK instruction or operation code trap, do not use the RETI instruction.

18.6.13 Stack handling instruction

PUSH sfrp

Function: (SP-1) ← sfr_H, (SP-2) ← sfr_L, SP ← SP-2
Save the contents of the special function register (register that can be handled in 16-bit units) in the stack memory addressed by the stack pointer (SP), then decrement the SP.

Flag operation: No change

PUSH post

Function: {(SP-1) ← $post_H$, (SP-2) ← $post_L$, SP ← SP-2} x n
(n is the number of register pairs described as post)
Save the contents of the 16-bit register pair specified in the operand in the stack memory, then decrement the SP.
More than one register pair name can be described in the operand post.
The save operation is performed starting at the register pair assigned to bit 7 of the 8-bit immediate data in the second byte (Post Byte).
The upper half of the register pair is saved in the stack addressed by (SP-2n+1) and the lower half is saved in the stack addressed by (SP-2n).

Flag operation: No change

PUSH PSW

Function: $(SP-1) \leftarrow PSW_H$, $(SP-2) \leftarrow PSW_L$, $SP \leftarrow SP-2$
Save the program status word (PSW) contents in the stack memory addressed by the stack pointer (SP) and decrement the SP.

Flag operation: No change

PUSHU post

Function: $\{(UP-1) \leftarrow post_H$, $(UP-2) \leftarrow post_L$, $UP \leftarrow UP-2\} \times n$
(n is the number of register pairs described as post)
Save the contents of the 16-bit register pair specified in the operand in the memory addressed by the user stack pointer (UP), then decrement the UP. More than one register pair name can be described in the operand post.
The save operation is performed starting at the register pair assigned to bit 7 of the 8-bit immediate data in the second byte (Post Byte).
The upper half of the register pair is saved in the memory addressed by (UP-2n+1) and the low half is saved in the stack addressed by (UP-2n).

Flag operation: No change

POP sfrp

Function: $sfr_L \leftarrow SP$, $sfr_H \leftarrow (SP+1)$, $SP \leftarrow SP+2$
Restore the contents of the stack memory addressed by the stack pointer (SP) to the special function register (register that can be handled in 16-bit units), then increment the SP.

Flag operation: No change

POP post

Function: {$post_L$ ← (SP), $post_H$ ← (SP+1), SP ← SP+2} x n
(n is the number of register pairs described as post)
Restore the contents of the stack memory addressed by the stack pointer (SP) to the 16-bit register pair specified in the operand, then increment the SP.
More than one register pair name can be described in the operand post.
The transfer operation is performed starting at the register pair assigned to bit 0 of the 8-bit immediate data in the second byte (Post Byte). The contents of the stack addressed by (SP+2n-2) are restored to the lower half of the register pair and the contents of the stack addressed by (SP+2n-1) are restored to the upper half.

Flag operation: No change

POP PSW

Function: PSW_L ← (SP), PSW_H ← (SP+1), SP ← SP+2
Restore the contents of the stack memory addressed by the stack pointer (SP) to the program status word (PSW) and decrement the SP.

Flag operation:

S	Z	AC	P/V	CY
R	R	R	R	R

POPU post

Function: {$post_L$ ← (UP), $post_H$ ← (UP+1), UP ← UP+2} x n
(n is the number of register pairs described as post)
Restore the contents of the stack memory addressed by the user stack pointer (UP) in the register pair specified in the operand, then increment the UP.
More than one register pair name can be described in the operand post.
The transfer operation is performed starting at the register pair assigned to bit 0 of the 8-bit immediate data in the second byte (Post Byte). The contents of the stack addressed by (UP+2n-2) are restored to the lower half of the register pair and the contents of the stack addressed by (UP+2n-1) are restored to the upper half.

Flag operation: No change

MOVW SP, #word

Function: SP ← word word=0000H-FDFEH (any desired addresses)
word=FE00H-FFFEH (limited to even addresses)
Transfer the 16-bit immediate data specified in the second operand to the stack pointer (SP).

Flag operation: No change

MOVW SP, AX

Function: SP ← AX
Transfer the 16-bit register pair AX contents to the stack pointer (SP).
When AX contains 0000H-FDFEH, any desired address can be used, but when it contains FE00H-FEFEH, only even addresses can be used.

Flag operation: No change

MOVW AX, SP

Function: AX ← SP
Transfer the stack pointer (SP) contents to the 16-bit register pair AX.

Flag operation: No change

INCW SP

Function: SP ← SP+1
Increment the stack pointer (SP) contents by one.
If an interrupt is acknowledged when the address stored in the SP is an odd address of FE00H-FEFFH, an error may be caused. Therefore, be sure to set the stored address to an even address.

Flag operation: No change

DECW SP

Function: SP ← SP-1
Decrement the stack pointer (SP) contents by one.
If an interrupt is acknowledged when the address stored in the SP is an odd address of FE00H-FEFFH, an error may be caused. Therefore, be sure to set the stored address to an even address.

Flag operation: No change

18.6.14 Special instructions

CHKL sfr

Function: (Pin level) ∀ (signal level at prestage of output buffer)
Exclusive-OR the pin level with the signal level at the prestage of the output buffer and set the result in the flags S and Z.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

CHKLA sfr

Function: A ← (pin level) ∀ (signal level at presentage of output buffer)
Exclusive-OR the pin level with the signal level at the prestage of the output buffer and set the result in the A register.

Flag operation:

S	Z	AC	P/V	CY
x	x		P	

18.6.15 Unconditional branch instructions

BR !addr16

Function: PC ← addr16 addr16=0000H-FDFFH

Transfer the 16-bit immediate data specified in the operand to the program counter (PC) for a branch to the address addressed by the PC.
A branch can be taken to memory addresses 0000H-FDFFH.

Flag operation: No change

Caution: Instructions cannot be fetched from addresses FE00H-FFFFH. Do not describe any of the instructions in addr16.

Description example: BR BLK3;Branch to the address indicated by label BLK3.

BR rp1

Function: PC_H ← $rp1_H$, PC_L ← $rp1_L$
Transfer the contents of the 16-bit register pair specified in the operand to the program counter (PC) for a branch to the address addressed by the PC.
A branch can be taken to memory addresses 0000H-FDFFH.

Flag operation: No change

Caution: Instructions cannot be fetched from address FE00H-FFFFH. Do not set any of the addresses in rp1.

BR [rp1]

Function: $PC_H \leftarrow (rp1+1)$, $PC_L \leftarrow (rp1)$
Transfer the contents of the 2-byte area of the memory addressed by the contents of the 16-bit register pair specified in the operand to the program counter (PC) for a branch to the address addressed by the PC.
A branch can be taken to memory addresses 0000H-FDFFH.

Flag operation: No change

Caution: Instructions cannot be fetched from address FE00H-FFFFH. Do not set any of the addresses in rp1.

BR $addr16

Function: PC ← PC+2+jdisp addr16=(PC-126)- (PC+129)
Transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

18.6.16 Conditional branch instructions

BC $addr16
BL $addr16

Function: PC ← PC+2+jdisp if CY=1 addr16=(PC-126)-(PC+129)
When the carry flag is 1, transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BNC $addr16
BNL $addr16

Function: PC ← PC+2+jdisp if CY=0 addr16=(PC-126)-(PC+129)
When the carry flag is 0, transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BZ $addr16
BE $addr16

Function: PC ← PC+2+jdisp if Z=1 addr16=(PC-126)-(PC+129)
When the zero flag is 1, transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

Description example:

```
DEC 0FE20H
BZ  $ JMP  ; Decrement the contents of the memory
             addressed by FE20H by one. When the
             memory contains 0, branch to the address
             indicated by label JMP. (However the
             branch destination must range from "top
             address of the next instruction -128"
             to "top address +127".
```

BNZ $addr16
BNE $addr16

Function: PC ← PC+2+jdisp if Z=0 addr16=(PC-126)-(PC+129)
When the zero flag is 0, transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BV $addr16
BPE $addr16

Function: PC ← PC+2+jdisp if P/V=1 addr16=(PC-126)-(PC+129)
When the parity/overflow flag is 1, transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BNV $addr16
BPO $addr16

Function: PC ← PC+2+jdisp if P/V=0 addr16=(PC-126)-(PC+129)
When the parity/overflow flag is 0, transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BN $addr16

Function: PC ← PC+2+jdisp if S=1 addr16=(PC-126)-(PC+129)
When the sign flag is 1, transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BP $addr16

Function: PC ← PC+2+jdisp if S=0 addr16=(PC-126)-(PC+129)
When the sign flag is 0, transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BGT $addr16

Function: PC ← PC+3+jdisp if (P/V ⊻ S) V Z=0
addr16=(PC-125)-(PC+130)
Exclusive-OR the parity/overflow flag contents with the sign flag contents and OR the result with the zero flag contents. When the result of ORing is 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

Description example: CMP A, #0FH
BGT $MR2 ;Branch to the address indicated by label MR2 if the two's complement data in the A register is greater than FH.

BGE $addr16

Function: PC ← PC+3+jdisp if P/V ⊻ S=0

addr16=(PC-125)-(PC+130)

Exclusive-OR the parity/overflow flag contents with the sign flag contents. When the result is 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BLT $addr16

Function: PC ← PC+3+jdisp if P/V ⊻ S=1

addr16=(PC-125)-(PC+130)

Exclusive-OR the parity/overflow flag contents with the sign flag contents. When the result is 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BLE $addr16

Function: PC ← PC+3+jdisp if (P/V ⊻ S) V Z=1

addr16=(PC-125)-(PC+130)

Exclusive-OR the parity/overflow flag contents with the sign flag contents and OR the result with zero flag contents. When the result of ORing is 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BH $addr16

Function: PC ← PC+3+jdisp if Z V CY=0

addr16=(PC-125)-(PC+130)

OR the zero flag contents with the carry flag contents. When the result is 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BNH $addr16

Function: PC ← PC+3+jdisp if Z V CY=1

addr16=(PC-125)-(PC+130)

OR the zero flag contents with the carry flag contents. When the result is 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BT saddr. bit, $addr16

Function: PC ← PC+3+jdisp if (saddr.bit) =1
addr16=(PC-125)-(PC+130)
saddr=FE20H-FF1FH
bit=0-7

When the short direct memory bit addressed in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the short direct memory bit address or label in the first operand saddr.bit as it is and the branch destination address directly with a label or numeric value in the second operand addr16 by considering the branch range.

Flag operation: No change

BT sfr.bit, $addr16

Function: PC ← PC+4+jdisp if sfr.bit=1
addr16=(PC-124)-(PC+131)
bit=0-7

When the bit of the special function register specified in the first operand, addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the fourth byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the second operand addr16 by considering the branch range.

Flag operation: No change

BT A.bit, $addr16

Function: PC ← PC+3+jdisp if A.bit=1
addr16=(PC-125)-(PC+130)
bit=0-7

When the A register bit addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the second operand addr16 by considering the branch range.

Flag operation: No change

Description example: BT A.3, $JMP1;Branch to the address indicated by label JMP1 if A register bit 3 is "1".

BT X.bit, $addr16

Function: PC ← PC+3+jdisp if X.bit=1
addr16=(PC-125)-(PC+130)
bit=0-7

When the X register bit addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BT PSWH.bit, $addr16

Function: PC ← PC+3+jdisp if PSW_H.bit=1
addr16=(PC-125)-(PC+130)
bit=0-7

When the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BT PSWL.bit, $addr16

Function: PC ← PC+3+jdisp if PSW_L.bit=1
addr16=(PC-125)-(PC+130)
bit=0-7

When the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BF saddr.bit, $addr16

Function: PC ← PC+4+jdisp if (saddr.bit) =0
addr16=(PC-124)-(PC+131)
saddr=FE20H-FF1FH
bit=0-7

When the short direct memory bit addressed in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the fourth byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the short direct memory bit address or label in the first operand saddr.bit as it is and the branch destination address directly with a label or numeric value in the second operand addr16 by considering the branch range.

Flag operation: No change

BF sfr.bit, $addr16

Function: PC ← PC+4+jdisp if sfr.bit=0
addr16=(PC-124)-(PC+131)
bit=0-7

When the bit of the special function register specified in the first operand, addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the fourth byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BF A.bit, $addr16

Function: PC ← PC+3+jdisp if A.bit=0
addr16=(PC-125)-(PC+130)
bit=0-7

When the A register bit addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BF X.bit, $addr16

Function: PC ← PC+3+jdisp if X.bit=0

addr16=(PC-125)-(PC+130)

bit=0-7

When the X register bit addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BF PSWH.bit, $addr16

Function: PC ← PC+3+jdisp if PSW_H.bit=0
addr16=(PC-125)-(PC+130)
bit=0-7

When the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BT PSWL.bit, $addr16

Function: PC ← PC+3+jdisp if PSW_L.bit=0
addr16=(PC-125)-(PC+130)
bit=0-7

When the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BTCLR saddr.bit, $addr16

Function: PC ← PC+4+jdisp if (saddr.bit) =1
then clear
addr16=(PC-124)-(PC+131)
saddr=FE20H-FF1FH
bit=0-7

When the short direct memory bit addressed in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the fourth byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the short direct memory bit address or label in the first operand saddr.bit as it is and the branch destination address directly with a label or numeric value in the second operand addr16 by considering the branch range.

Flag operation: No change

BTCLR sfr.bit, $addr16

Function: PC ← PC+4+jdisp if sfr.bit=1
then clear
addr16=(PC-124)-(PC+131)
bit=0-7

When the bit of the special function register specified in the first operand, addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from 8-bit displacement value jdisp in the fourth byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BTCLR A.bit, $addr16

Function: PC ← PC+3+jdisp if A.bit=1
then clear
addr16=(PC-125)-(PC+130)
bit=0-7

When the A register bit addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then, reset the bit to 0.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BTCLR X.bit, $addr16

Function: PC ← PC+3+jdisp if X.bit=1
then clear
addr16=(PC-125)-(PC+130)
bit=0-7

When the X register bit addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then, reset the bit to 0.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BTCLR PSWH.bit, $addr16

Function: PC ← PC+3+jdisp if PSW_H.bit=1
then clear
addr16=(PC-125)-(PC+130)
bit=0-7

When the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then, reset the bit to 0.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BTCLR PSWL.bit, $addr16

Function: PC ← PC+3+jdisp if PSW_L.bit=1
then clear
addr16=(PC-125)-(PC+130)
bit=0-7

When the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand is set to 1, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then, reset the bit to 0.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: If the specified flag is "1", it is reset to 0.

Description example: BTCLR PSWL.3, $0F6EH;Reset UF flag to 0 if the flag is "1" and branch to address F6EH.

BFSET saddr.bit, $addr16

Function: PC ← PC+4+jdisp if (saddr.bit) =0
then set
addr16=(PC-124)-(PC+131)
saddr=FE20H-FF1FH
bit=0-7

When the short direct memory bit addressed in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the fourth byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then set the addressed bit to 1.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BESET sfr.bit, $addr16

Function: PC ← PC+4+jdisp if sfr.bit=0
then set
addr16=(PC-124)-(PC+131)
bit=0-7

When the bit of the special function register specified in the first operand, addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the fourth byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then, set the addressed bit to 1.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BFSET A.bit, $addr16

Function: PC ← PC+3+jdisp if A.bit=0
then set
addr16=(PC-125)-(PC+130)
bit=0-7

When the A register bit addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then, set the addressed bit to 1.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BFSET X.bit, $addr16

Function: PC ← PC+3+jdisp if X.bit=0
then set
addr16=(PC-125)-(PC+130)
bit=0-7

When the X register bit addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then, set the addressed bit to 1.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BFSET PSWH.bit, $addr16

Function: PC ← PC+3+jdisp if PSW_H.bit=0
then set
addr16=(PC-125)-(PC+130)
bit=0-7

When the bit of the high-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then, set the addressed bit to 1.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

BFSET PSWL.bit, $addr16

Function: PC ← PC+3+jdisp if PSW_L.bit=0
then set
addr16=(PC-125)-(PC+130)
bit=0-7

When the bit of the low-order eight bits of the program status word (PSW), addressed by the 3-bit immediate data in the first operand is set to 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC. Then, set the addressed bit to 1.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.
Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: If the specified flag is "0", it is set to 1.

DBNZ r2, $addr16

Function: r2 ← r2-1, then PC ← PC+2+jdisp if r2≠0
addr16=(PC-126)-(PC+129)

Decrement the contents of the 8-bit register specified in the first operand by one. If the result is not 0, transfer the value resulting from adding 8-bit displacement value jdisp in the second byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.
jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

DBNZ saddr, $addr16

Function: (saddr) ← (saddr)-1 then PC ← PC+3+jdisp is (saddr)≠0

addr16=(PC125)-(PC+130)

saddr=FE20H-FF1FH

Decrement the contents of the short direct memory specified in the first operand by one. If the result is not 0, transfer the value resulting from adding 8-bit displacement value jdisp in the third byte of the operation code to the top address of the next instruction to the program counter (PC) for a branch to the address addressed by the PC.

jdisp is handled as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

Describe the branch destination address directly with a label or numeric value in the operand addr16 by considering the branch range.

Flag operation: No change

18.6.17 Context switching instruction

BRKCS RBn

Function: RBS2-0 ← n, PC_H ← R5, PC_L ← R4, R7 ← PSW_H, R6 ← PSW_L, RSS ← 0, IE ← 0 n=0-7

Set 3-bit immediate data N_{2-0} in the operation code in the register bank selection flag (RBS2-0) to select register bank n described in the operand. Exchange the contents of the 8-bit registers R5 and R4 of the register bank and the program counter (PC) and save the program status word (PSW) contents in the 8-bit register R7 and R6 for a branch to the address set in R5 and R4. Then, reset the RSS and IE flags to 0.

Flag operation: No change

RETCS !addr16

Function: PC_H ← R5, PC_L ← R4, R5 ← $addr16_H$, R4 ← $addr16_L$, PSW_H ← R7, PSW_L ← R6

addr16=0000H-FDFFH

Transfer the contents of 8-bit register R7 to R4 in the register bank specified during execution of the RETCS instruction to the program status word (PSW) and program counter (PC) and return to the address set in R5 and R4. Then, transfer the 16-bit immediate data specified in the operand to R5 and R4.

The RETCS instruction is effective for context switching when an interrupt request occurs; it is used to return from context switching branch instruction. addr16 described in the operand becomes branch destination address when the same register bank is again specified by the context switching function.

Flag operation:

S	Z	AC	P/V	CY
R	R	R	R	R

Caution 1: Instructions cannot be fetched from address FE00H-FFFFH. Do not describe any of the addresses in addr16.

2: To return from BRKCS instruction branch processing, do not use the RETCS instruction.

RETCSB !addr16

Function: $PC_H \leftarrow R5$, $PC_L \leftarrow R4$, $R5 \leftarrow addr16_H$, $R4 \leftarrow addr16_L$, $PSW_H \leftarrow R7$, $PSW_L \leftarrow R6$

addr16=0000H-FDFFH

Transfer the contents of 8-bit register R7 to R4 in the register bank specified during execution of the RETCSB instruction to the program status word (PSW) and program counter (PC).
Then, return to the address set in R5 and R4.
The BRKCSB instruction is used to return from the BRKCS instruction (See 13.5).

Flag operation:

S	Z	AC	P/V	CY
R	R	R	R	R

Caution: To return from the interrupt service routine started by executing the BRKCS instruction, be sure to use the RETCSB instruction.
If the RETCS instruction is used, the interrupt control circuit does not operate normally.

18.6.18 String instructions

MOVM [DE+],A
MOVM [DE-],A

Function: {(DE) ← A, DE ← DE+1/-1, C ← C-1}
End if C=0
Transfer the A register contents to the memory addressed by the register pair DE and increment/decrement the register pair DE contents by one. Then, decrement the C register contents by one. Repeat these steps until the C register contains 0.

Flag operation: No change

Description example:

```
MOV   R2, #00H      ; C  ← 0H
MOV   R1, #00H      ; A  ← 00H
MOVW RP6, #FE00H    ; DE ← FE00H
MOVM [DE+],A        ; Clear RAM of FE00H-FEFFH.
```

MOVBK [DE+], [HL+]
MOVBK [DE-], [HL-]

Function: {(DE) ← (HL), DE ← DE+1/-1, HL ← HL+1/-1,
C ← C-1}
End if C=0
Transfer the contents of the memory addressed by the register pair HL to the memory addressed by the register pair DE and increment/decrement the register pair DE and HL contents by one. Then, decrement the C register contents by one. Repeat these steps until the C register contains 0.

Flag operation: No change

Description example:

```
MOV  R2,  #10H      ; C  ← 10H
MOVW RP6, #3000H    ; DE ← 3000H
MOVW RP7, #5000H    ; HL ← 5000H
MOVBK [DE+],[HL+]   ; Transfer the contents of memory
                      addresses 5000H-500FH to memory
                      addresses 3000H-300FH.
```

XCHM [DE+],A
XCHM [DE-],A

Function: {(DE) ←→ A, DE ← DE+1/-1, C ← C-1}
End if C=0
Exchange the contents of the A register and the memory addressed by the register pair DE and increment/decrement the register pair DE contents by one. Then, decrement the C register contents by one. Repeat these steps until the C register contains 0.

Flag operation: No change

Description example:

```
MOV   R2, #10H      ; C  ← 10H
MOV   R1, #00H      ; A  ← 00H
MOVW RP6, #3050H    ; DE ← 3050H
XCHM [DE+],A          ; Shift the contents of memory
                      addresses 3050H-305FH backward
                      one address (Address 3050H
                      contents are set to 0.)
```

XCHBK [DE+], [HL+]
XCHBK [DE-], [HL-]

Function: {(DE) ⟷ (HL), DE ← DE+1/-1, HL ← HL+1/-1, C ← C-1}
End if C=0
Exchange the contents of the memory addressed by the register pair HL to the memory addressed by the register pair DE and increment/decrement the register pair DE and HL contents by one. Then, decrement the C register contents by one. Repeat these steps until the C register contains 0.

Flag operation: No change

CMPME [DE+],A
CMPME [DE-],A

Function: {(DE)-A, DE ← DE+1/-1, C ← C-1}
End if C=0 or Z=0
Compare the A register contents with the contents of the memory addressed by the register pair DE, increment/decrement the register pair DE contents by one, and decrement the C register contents by one.
Repeat these steps until a mismatch is found as a result of the comparison or the C register contains 0.
Execution of the instruction does not affect the A register contents or the contents of the memory addressed by the register pair DE.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPBKE [DE+], [HL+]
CMPBKE [DE-], [HL-]

Function: {(DE)-(HL), DE ← DE+1/-1, HL ← HL+1/-1, C ← C-1}
End if C=0 or Z=0
Compare the contents of the memory addressed by the register pair HL with the contents of the memory addressed by the register pair DE, increment/decrement the register pair DE and HL contents by one, and decrement the C register contents by one. Repeat these steps until a mismatch is found as a result of the comparison or the C register contains 0.
Execution of the instruction does not affect the contents of the memory locations addressed by the register pairs HL and DE.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPMNE [DE+],A
CMPMNE [DE-],A

Function: {(DE)-A, DE ← DE+1/-1, C ← C-1}
End if C=0 or Z=1
Compare the A register contents with the contents of the memory addressed by the register pair DE, increment/decrement the register pair DE contents by one, and decrement the C register contents by one.
Repeat these steps until a match is found as a result of the comparison or the C register contains 0.
Execution of the instruction does not affect the A register contents or the contents of the memory addressed by the register pair DE.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Description example:

```
MOV   R2, #00H      ; C   ← 00H
MOVW RP6, #3000H    ; DE  ← 3000H
CMPMNE[DE+], A
BZ                  ; Branch to the address indicated
                      by label JMP if any of address-
                      es 3000H-30FFH contains the
                      same value as the A register.
```

CMPBKNE [DE+], [HL+]
CMPBKNE [DE-], [HL-]

Function: {(DE)-(HL), DE ← DE+1/-1, HL ← HL+1/-1, C ← C-1}
End if C=0 or Z=1
Compare the contents of the memory addressed by the register pair HL with the contents of the memory addressed by the register pair DE, increment/decrement the register pair DE and HL contents by one, and decrement the C register contents by one. Repeat these steps until a match is found as a result of the comparison or the C register contains 0.
Execution of the instruction does not affect the contents of the memory locations addressed by the register pairs HL and DE.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPMC [DE+],A
CMPMC [DE-],A

Function: {(DE)-A, DE ← DE+1/-1, C ← C-1}
End if C=0 or CY=0
Compare the A register contents with the contents of the memory addressed by the register pair DE, increment/decrement the register pair DE contents by one, and decrement the C register contents by one.
Repeat these steps until the contents of the memory addressed by the register pair DE becomes greater than the A register contents as a result of the comparison or the C register contains 0.
Execution of the instruction does not affect the A register contents or the contents of the memory addressed by the register pair DE.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPBKC [DE+], [HL+]
CMPBKC [DE-], [HL-]

Function: {(DE)-(HL), DE ← DE+1/-1, HL ← HL+1/-1, C ← C-1}
End if C=0 or CY=0
Compare the contents of the memory addressed by the register pair HL with the contents of the memory addressed by the register pair DE, increment/decrement the register pair DE and HL contents by one, and decrement the C register contents by one. Repeat these steps until the contents of the memory addressed by the register pair DE become greater than the contents of the memory addressed by the register pair HL as a result of the comparison or the C register contains 0.

Execution of the instruction does not affect the contents of the memory locations addressed by the register pairs HL and DE.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

CMPMNC [DE+],A
CMPMNC [DE-],A

Function: {(DE)-A, DE ← DE+1/-1, C ← C-1}
End if C=0 or CY=1
Compare the A register contents with the contents of the memory addressed by the register pair DE, increment/decrement the register pair DE contents by one, and decrement the C register contents by one.
Repeat these steps until the A register contents become greater than the contents of the memory addressed by the register pair DE as a result of the comparison or the C register contains 0.
Execution of the instruction does not affect the A register contents or the contents of the memory addressed by the register pair DE.

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

Description example:

```
MOV     R2, #00H        ; C   ← 00H
MOVW    RP6, #8000H     ; DE  ← 8000H
CLR1    CY              ; CY  ← 0
CMPMNC  [DE+], A
BC      $JMP            ; Branch to the address indicated
                          by label JMP if any of addresses 8000H-80FFH contains a greater value than the A register.
```

CMPBKNC [DE+], [HL+]
CMPBKNC [DE-], [HL-]

Function: {(DE)-(HL), DE ← DE+1/-1, HL ← HL+1/-1, C ← C-1}
End if C=0 or CY=1
Compare the contents of the memory addressed by the register pair HL with the contents of the memory addressed by the register pair DE, increment/decrement the register pair DE and HL contents by one, and decrement the C register contents by one. Repeat these steps until the contents of the memory addressed by the register pair HL become greater than the contents of the memory addressed by the register pair DE as a result of the comparison or the C register contains 0.
Execution of the instruction does not affect the contents of the memory locations addressed by the register pairs HL and DE

Flag operation:

S	Z	AC	P/V	CY
x	x	x	V	x

18.6.19 CPU control instructions

MOV STBC, #byte

Function: STBC ← byte byte=00H-FFH
Set the 8-bit immediate data specified in the second operand in the standby control register (STBC).
This instruction has a special operation code to set the STBC register.

Flag operation: No change

Description example: MOV STBC, #01H,Set HALT mode.

MOV WDM, #byte

Function: WDM ← byte byte=00H-FFH
Set the 8-bit immediate data specified in the second operand in the watchdog timer mode register (WDM).
This instruction has a special operation code to set the WDM register.

Flag operation: No change

SWRS

Function: RSS ← $\overline{RSS}$
Invert the register set selection flag (RSS) contents.

Flag operation: No change

SEL RBn

Function: RBS2-0 ← n, RSS ← 0 n=0-7
Set the 3-bit immediate data in the operation code (N_{0-2}) in the register bank selection flag (RBS0-RBS2) to select the register bank described in the operand. Reset the register set selection flag (RSS) to 0.

Flag operation: No change

SEL RBn, ALT

Function: RBS2-0 ← n, RSS ← 1 n=0-7
Set the 3-bit immediate data in the operation code (N_{0-2}) in the register bank selection flag (RBS0-RBS2) to select the register bank described in the operand. Set the register set selection flag (RSS) to 1.

Flag operation: No change

NOP

Function: No Operation
Perform no operation and consume two clocks.

Flag operation: No change

EI

Function: IE ← 1
Set the interrupt request enable flag (IE) to 1. Whether maskable interrupt acknowledge is enabled or disabled is controlled by setting the interrupt request control registers.

Flag operation: No change

DI

Function: IE ← 0
Reset the interrupt request enable flag (IE) to 0 to disable acknowledge of every maskable interrupt.

Flag operation: No change

CHAPTER 4 BLOCK FUNCTION OUTLINE

4.1 Execution Unit

The execution unit (EXU) executes address calculation, arithmetic and logical operations, data transfer, etc., under the microprogram control.

The EXU contains 256-byte main RAM. Eight register banks are mapped in the EXU internal main RAM.

4.2 Bus Control Unit

The bus control unit (BCU) starts necessary bus cycles based on the physical address provided by the execution unit (EXU). When the EXU does not request bus cycle start, the BCU generates an address for prefetch and fetches a given instruction. The code of the prefetched instruction is read into a 3-byte instruction queue.

4.3 Program Memory/Data Memory

This block consists of 32K-byte program memory (ROM) and 768-byte data memory (peripheral RAM). However, the uPD78330 does not contain the ROM.

If the $\overline{EA}$ pin of the uPD78334, 78P334 is fixed low, an access to the uPD78334 internal mask ROM, PROM can be inhibited.

4.4 Ports

Table 4-1 lists the uPD78334 port types.

Every port can be handled bit-wise as well as in 8-bit units for extremely versatile control. In addition to digital port operation, the ports function of on-chip hardware input/output pins as dual function.

Table 4-1 Port Function and Dual Function

Port name	Port function	Dual function
Port 0	8-bit input/output port. Input or output mode can be specified bit-wise.	Real-time output port (RTP) in control mode.
Port 1	8-bit input/output port. Input or output mode can be specified bit-wise.	Real-time pulse unit (RPU) output in control mode.
Port 2	8-bit input-only port.	External interrupt input and real-time pulse unit (RPU) capture trigger input and count pulse input in control mode.
Port 3	8-bit input/output port. Input or output mode can be specified bit-wise.	Serial interface (UART or CSI) input/output and real-time pulse unit (RPU) output in control mode
Port 4(Note 1)	8-bit input/output port. Input or output mode can be specified in 8-bit units.	Address/data bus (AD0-AD7) when memory is expanded
Port 5(Note 1)	8-bit input/output port. Input or output mode can be specified bit-wise.	Address bus (A8-A15) when memory is expanded.
Port 7	8-bit input-only port.	A/D converter analog input in control mode
Port 8	8-bit input-only port.	A/D converter analog input in control mode.
Port 9(Note 2)	6-bit input/output port. Input or output mode can be specified bit-wise.	Control signal output when memory is expanded is connected. PWM signal output in control mode.

Note 1: uPD78330 ports 4 (P4) and 5 (P5) function only as address/data bus and address bus respectively.

Note 2: The low-order two bits of port 9 (P9) function as the RD and $\overline{WR}$ signals. When the uPD78334 $\overline{EA}$ pin is fixed low, the pins also function as the $\overline{RD}$ and $\overline{WR}$ signals.

4.5 Real-Time Pulse Unit (RPU)

The real-time pulse unit (RPU) consists of the following hardware devices:

- 18/16-bit timer/counter x 1
  18/16-bit compare registers x 5
  18/16-bit capture registers x 3
  18/16-bit capture/compare registers x 2
  pulse output lines x 6
- 16-bit timer/counter x 3
  16-bit compare registers x 5
  16-bit capture register x 1
  timer output pins x 5
- Real-time output port x 8

The RPU can perform programmable pulse output and pulse interval and frequency measurements.

The greatest feature of the real-time pulse unit lines in rich and multifunctional timer pulse output. A total of six timer pulse output lines such as toggle output, set output, and reset output can be controlled independently. In addition, the real-time output port output timing can be controlled.

The set and reset timings of real-time output port output are also controlled.

4.6 A/D Converter

High-speeds, high-resolution 10-bit A/D (analog/digital) converter is contained.

16 analog input lines (ANI0-ANI15) are contained and various functions are provided conforming to application such as the select mode, scan mode, and mix mode.

4.7 Serial Interface

Two independent serial interface channels of asynchronous serial interface (UART) and clocked serial interface are provided. Also, a baud rate generator common to both the channels is contained.

On the asynchronous serial interface, data is transferred through the TxD and RxD pins.

The clocked serial interface has the following two operation modes:

- 3-line serial I/O mode
  Data is transferred through the three pins of serial clock ($\overline{SCK}$), serial input (SI), and serial output (SO).
- Serial bus interface (SBI) mode
  Data is transferred through the two pins of serial clock ($\overline{SCK}$) and serial data bus (SB0 or SB1).

4.8 Watchdog Timer

The watchdog timer is a free-running timer which has the nonmaskable interrupt function to prevent software upset or deadlock. A program error can be known by the fact that a watchdog timer overflow interrupt (INTWDT) occurs or the watchdog timer output pin ($\overline{WDTO}$) goes low. If the output is connected to the $\overline{RESET}$ pin, a program error can be prevented from causing an application system error.

4.9 PWM Output Unit

Two 8-bit precision PWM signal outputs are contained. PWM output can be used as digital-analog conversion output by connecting an external low-pass filter, etc. It is appropriate for actuator control signals of motors, etc.

4.10 Interrupt Controller

The interrupt controller handles various interrupt requests occurring from the peripheral hardware and the external (NMI and INTP0-INTP6) in any mode of context switching, vectored interrupt, and macro service. In addition, three interrupt priority levels are programmable by the software.